AUTOMATIC GENERATION OF FLEXIBLE CMOS CELLS FOR GENERAL CELL VLSI LAYOUT SYSTEMS

By

Wanhao Li

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1987

ACKNOWLEDGEMENTS

The author wishes to express his gratitude to Professor Julius T. Tou, the chairman of committee, for all the technical guidance, innovative discussion, and continuous support throughout his graduate studies. The author would also like to extend his deep appreciation to Dr. John Staudhammer and Dr. Yuan-Chieh Chow for their kind help in all areas. Special thanks are due to Dr. Fred J. Taylor for the special permission to use his research equipment. Special thanks are also due to Dr. Mark C. Yang for his valuable suggestion in research methodologies.

Special thanks are given to all his colleagues at the Center for Information Research for their encouragement and helpful discussions. Thanks are also due to a good friend, Mr. Jing-Jang Hwang, for his valuable discussion and kind assistance.

The author wants to thank his parents and family, especially his mother Chen-Jin Li, for their love, expectation, and encouragement.

The author also wishes to express his deep appreciation to his wife Yarly for all the patience, encouragement, and great help. He also wants to thank his lovely daughter Jar-Shing for bringing such happiness.

Finally, the author wishes to express his special gratefulness to the Center for Information Research. Without the continuous financial support from the center, this research study would have never been completed.

TABLE OF CONTENTS

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

AUTOMATIC GENERATION OF FLEXIBLE CMOS CELLS FOR GENERAL CELL VLSI LAYOUT SYSTEMS

By

Wanhao Li

August 1987

Chairman: Dr. Julius T. Tou
Major Department: Electrical Engineering

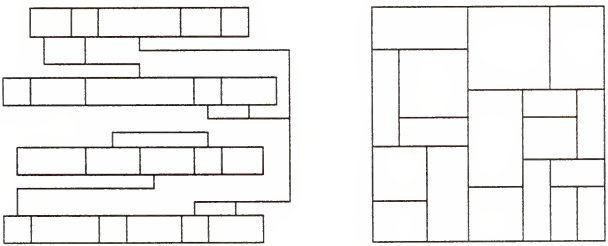
Cell-library design approaches have been the most popular VLSI design methods for the past decade. General cell layout, which uses cells with different shapes and sizes, is the cell-library design approach widely used for high-complexity, high-density VLSI layout design. However, one major drawback of this design approach is that the shapes and sizes of the prestored cells usually do not fit well into the allotted chip subareas. Such kind of cell shape mismatch causes sizable redundant chip area and requires laborious cell redesign work to improve it. In order to alleviate the problem, an automatic, flexible cell layout generation method has been studied and developed. The developed method uses delicate cell structure modeling and layout techniques to produce a custom-fit symbolic leaf cell from a transistor connectivity description and a description of the environment in which the cell will reside. The cell generation system not only produces good quality cell layouts within short turn-around-time but also reduces redundant chip area with custom-fit cells. Consequently, the function of the general cell layout system is greatly enhanced.

CHAPTER I
INTRODUCTION

With the advance of IC technology, the chip complexity has grown to the order of hundred thousands gates level. Consequently, the difficulty of integrated circuit design has increased exponentially. Design and layout for IC packages have become so time-consuming that it usually requires a few years and several good designers to develop a state-of-the-art custom VLSI chip. In order to reduce the chip turn-around-time and the closely related chip design cost, numerous systematic design methodologies and automatic layout techniques have been proposed.

Polycell layout (see Figure 1.1(a)), or called standard cell layout, was the first cell-based automatic layout method proposed in early 1970s. It has been successfully used in many automated layout systems, including MP2D [1], LTX [2], and CALMOS [3]. In most such systems, all cells are prepared to have the same height so as to simplify the layout structure and to reduce the layout design time. Unfortunately, this kind of constraint considerably limits the range of cell complexity and creates interconnection problems for complicated cells [4].

General cell layout (see Figure 1.1(b)), or called building block layout, has been discussed for years as an alternative approach for cell-based automatic layout. In such systems, cells are allowed to have different shapes and sizes which alleviates the problems created by polycell layout system. By using a hierarchical approach [5], general cell layout systems can handle chip design with very high complexity. In addition, general cell layout systems usually produce



(a) (b)

Figure 1.1 Sample Chip Floor Plans. a) Polycell Layout; b) General Cell Layout

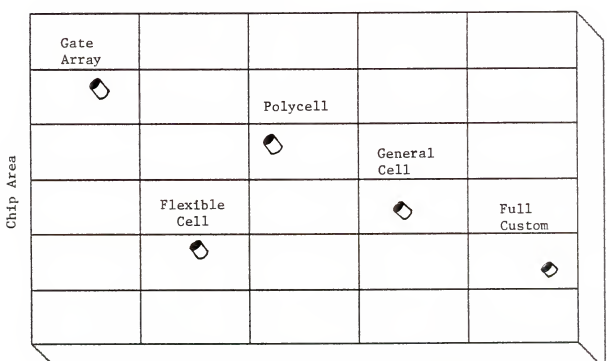
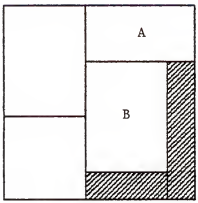


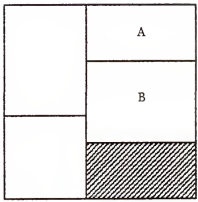
Figure 1.2 Comparison of the Layout Performance of Different Approaches.

chips with smaller sizes compared with polycell layout systems. The superiority in chip size for general cell layout system is demonstrated in Figure 1.2. Although the chip size is still much larger compared with the full-custom layout of random-logic networks, it is about 50% smaller than the size of the chips generated by polycell layout systems [6].

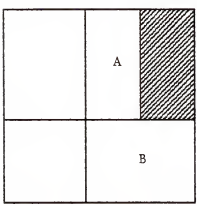
Although the general cell layout method is obviously superior in reducing chip area, some major drawbacks have severely limited its function in practical use. In most of the current general cell layout systems, only the block-level layout (chip floor plan) is automated. Functional blocks (cells) are manually designed and prestored in the cell library for the use of chip floor planning process. Since the shapes of all the cells are fixed beforehand, sizable redundant chip area is caused by shape mismatch between neighboring cells. Figure 1.3(a) is an example which shows that the shape mismatch between cell A and cell B causes redundant chip area which is marked by hatched lines. To reduce such redundant area, researchers have proposed placement improvement techniques such as rotation, squeezing, reflecting [7], and block packing [8]. However, the block-level placement improvement techniques are constrained by the initial floor plan structure and the fixed cell configurations, and can only be applied to limited cases. Figures 1.3(b) to 1.3(d) show that the redundant chip area cannot be reduced by any of the block-level techniques in Lauther [8]. In order to resolve the problem, configurations of the cells have to be made flexible. Figure 1.3(e) shows that the redundant chip area in Figure 1.3(a) is eliminated by reshaping cells A to a different configuration. In the past, such cell redesign work was done by experienced designers. This work was very time consuming because multiple iterations were usually required for a good chip layout design. In terms of this need, an automatic, flexible cell layout generation system is considered as indispensable in enhancing the function of the current general cell



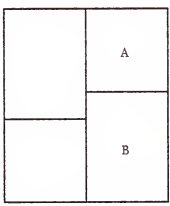
(a)



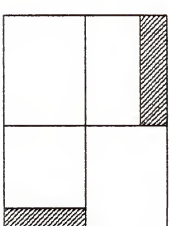
(d)



(b)



(e)



(c)

Figure 1.3 Improve the Redundant Chip Area Caused by Cell Shape Mismatch.
a) Original Chip Floor Plan; b) Rotate Cell A and B;
c) Rotate Cell A; d) Rotate Cell B; e) Reconfigure Cell A.

layout systems. The work presented in this dissertation pursues this research direction.

There are two main objectives sought in this dissertation. The first objective is to study the interactive behavior between the chip floor plan and cell design. The problem of cell design is considered under the global chip design environment. The second objective is to transcend the automatic cell layout schemes of the past and develop new layout techniques that lead to improved layouts of the cells which meet the requirements of the chip floor plan.

The design of the cell layout generation system is based upon one of the symbolic layout methods [9-14] -- Gate Matrix Layout [9]. Layout generation is executed on a virtual grid structure with equally spaced grids. Topology of the layout primitives, i.e. devices and wires, is determined by graph-theoretical and heuristic algorithms and is recorded as a set of symbols located on the grid structure. The symbolic layout is then compacted by cell compactors [15-19] to generate physical layout. Because most of the enumeration and computation are performed at discrete and symbolic levels, the cell design time is greatly reduced. Moreover, the symbolic layout approach isolates the design rules from the layout algorithms and makes the layout technology independent. The cost of designing and maintaining the cell library is also highly reduced.

Chapter II presents a survey on several recent automatic symbolic layout generation algorithms. Performance and constraints for those algorithms are comparatively discussed.

Chapter III gives a complete overview of the automatic cell layout generation problem. The problem is considered under the whole VLSI layout environment. A flexible grid structure is proposed for cell layout such as to simplify the design and to increase the flexibility. Layout problems are then formulated based on the grid structure.

Chapter IV carries two major geometric analyses to determine the cell configurations. The chip floor plan analysis is used to reassign the cells with new chip subareas such that the shape mismatch between neighboring cells is minimized. The purpose of the cell boundary analysis is to choose a grid structure for the cell layout process such that the shape mismatch between the cell and its apportioned chip subarea is minimized.

Chapter V illustrates the device placement algorithms which determine the locations of the devices on the grid structure. Three grid locations of the devices are to be determined in this process. Task I uses a linear order algorithm to resolve the order of device columns and determines the horizontal grid locations of the devices at the same time. Task II employs several circuit heuristics to determine the relative positions of the source and drain terminals on each device. Task III utilizes a vertical constraint graph concept to determine the vertical grid locations of the devices.

In Chapter VI, a device wiring algorithm for cell layout is described. The objective of this process is to realize the circuit connectivity with a minimum number of wire segments and contacts. The process is also decomposed into three tasks. Terminal labeling is a task which labels the net number of each device terminal on its grid location. This task provides a data structure for the use of the other two routing tasks. All the grid points with the same net number constitute a multi-point net which should be connected by wire segments. The task of wire-list generation is then used to find a shortest path which connects each terminal of the multi-point net. The final task of device wiring, which is called detail routing, is then implemented to realize the generated shortest path with minimum number of wire segments and contacts.

Chapter VII summarizes the major accomplishments reported in this dissertation and provides suggestions for further research.

CHAPTER II
PREVIOUS RESEARCH ON AUTOMATIC CELL LAYOUT ALGORITHMS

During the past two decades, considerable efforts have been devoted toward replacing manual IC layout design work by computers. Consequently, many automatic layout algorithms have been reported. However, only a few algorithms [20-34] have been developed for automatic cell layout design. One of the major reasons for the lack of success in this area is that human visual inspection is usually better than CAD software in designing networks that match the shapes and sizes of usable subareas of the chip. Automatic generation of flexible cells for the use of general cell VLSI layout systems is a problem yet to be resolved.

A survey on the various research and development studies on automatic cell layout generation is given below. The success of an automatic cell layout algorithm relies on whether it can exploit the design search space with underlying modularity and geometric simplicity. On the contrary, the underlying layout structure should be flexible enough to allow the generated cells to have different configurations. The discussion of the layout algorithms will be focused on these two main themes: geometric regularity and geometric flexibility of the layout environment.

2.1 Automatic Cell Layout Algorithms Based on Weinberger's Layout Structure

Weinberger's layout array [20] was the first structured design methodology proposed for cell layout. In his method, each circuit was implemented as a one-dimensional array of NAND/NOR gates. Each NAND/NOR gate has an

output and one or several inputs and is realized as an enlongated vertical diffusion area with horizontal cross-over input signals. Input signals are connected with the NAND/NOR gates through the formation of input devices, which are formed between the NAND/NOR diffusion strip and a parallel ground diffusion strip. The connections of the output signals are formed by placing contacts between signals and NAND/NOR diffusion strips.

Ohtsuki et al. [21] were the earliest researchers to develop automatic layout algorithms based on Weinberger's layout structure. Their method modeled each circuit (see Figure 2.1) as a netlist structure (see Figure 2.2), where each column represents a NAND/NOR gate and the horizontal wires represent the between-gate connections. An interval graph is then formulated as shown in Figure 2.3. Each vertex of the interval graph indicates a net segment in Figure 2.2. An edge between two vertices indicates that the net segments represented by those two vertices are overlapped. Their problem is then formulated as finding a permutation of the gates of which the interval graph has the minimum clique number (the number of the vertices of a complete subgraph). Since the clique of the interval graph indicates the number of net segments which are overlapped and cannot be placed at the same track, a minimum number of cliques indicate a minimum number of horizontal tracks required for the placement of net segments. Consequently, the cell area is minimized. However, there are several drawbacks associated with this study. One is that the search for the minimum clique of any graph is a classical NP-complete problem [35]. Consequently, they had to rely on heuristic algorithms to resolve the problem. A near-optimum instead of an absolute-optimum result was derived. Another problem is that sizable redundant cell area (white space) is usually created by using gates instead of transistors as layout primitives. The generated cells are generally much larger than the manually designed cells. The

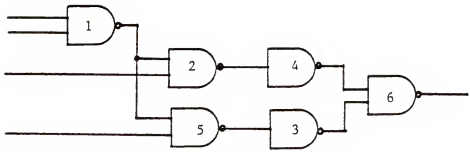


Figure 2.1 A Sample NAND Gate Logic Circuit.

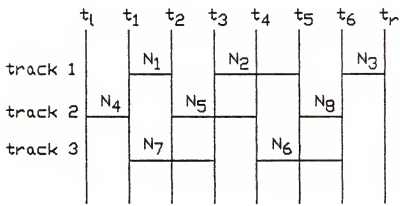


Figure 2.2 Netlist Representation of the Circuit in Figure 2.1.

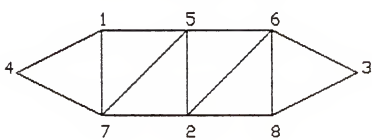


Figure 2.3 Interval Graph Representation Corresponding to the Placement in Figure 2.2.

other problem for this approach is that there is no topological flexibility for the generated cells. For a given circuit, the generated cell layout will have only one kind of configuration because a single-row layout structure is assumed. The generated cells, therefore, have very limited use in general cell layout systems or full-custom design.

Another work which used Weinberger's layout structure was developed by Uehara and vanCleemput [22]. In order to reduce white space within the cells, transistors were used as layout primitives in their algorithm. They also used CMOS technology to replace the original NMOS technology. In their approach, circuits were assumed to be constructed by the complementary switches circuit design method [36]. Each circuit is realized as series and parallel combinations of transistors where the p-structure was always the logical dual of the n-structure. Consequently, an CMOS transistor circuit can be converted to a special graph where 1) the vertices in the graph are the source/drain connections, and (2) the edges in the graph are transistors that connect particular source-drain vertices. Two graphs result, one for the n-logic tree and one for the p-logic tree. Figure 2.4 shows an example of the graph transformation. The connection of edges in the graphs mirrors the series-parallel connections in the circuit. If two edges are adjacent in the p- or n-graph, then they may share a common source-drain connection and may be connected by abutment. Furthermore, if there exists a sequence of edges in the p-graph and n-graph that have identical labeling, then the gate may be designed with no breaks. This path is known as an Euler path [35], which is defined as a closed walk running through every edge exactly once. The main points of the algorithm are as follows:

1. Find all Euler paths that cover the graph.
2. Find a p- and n- Euler path that have identical labeling (a labeling is an ordering of the gate labels on each vertex).

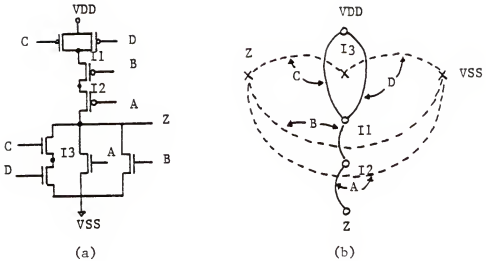


Figure 2.4 CMOS Logic Gate Graph Representation. a) A Sample Circuit; b) Graph Representation of a).

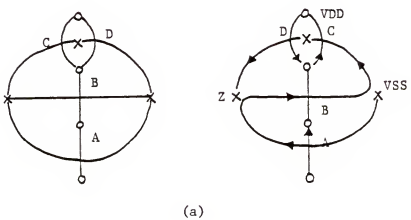


Figure 2.5 Layout Design by Euler Path. a) Euler Paths of Figure 2.4; b) The Corresponding Layout.

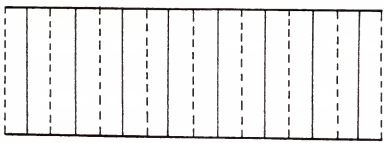
3. If (2) is not found, then break the gate in the minimum number of places to achieve (2) by separate Euler paths.

In this approach, cell area is minimized through the maximization of the number of transistor abutments. Figure 2.5 shows an example of cell layout design by Euler graph. However, there are two drawbacks for this method. First, the Euler path searching process is also an NP-complete problem which is difficult to be implemented. Second, their assumption of two-row layout structure also limits the flexibility of cell configurations.

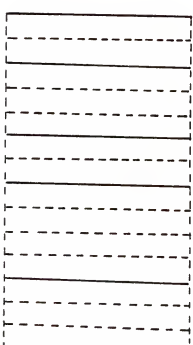
2.2 Algorithms Based on Gate Matrix Layout Structure

The Gate Matrix Layout method [9] is an evolution from Weinberger's layout methodology. It intended to provide a layout method which can generate compact CMOS cells. The idea of this method is to use transistors and wires as the layout primitives. An array of vertical polysilicon lines is used as both the sites of transistors and the connecting paths of the transistors. The transistors are placed on the polysilicon lines and are interconnected by three kinds of paths: vertical polysilicon lines, vertical diffusion runs, and horizontal metal wires. Diffusion columns are located between two polysilicon lines. Horizontal metal wires are placed on matrix rows together with the devices (transistors). The devices are placed horizontally with the vertical polysilicon lines crossing the gate terminal and diffusion columns crossing the drain and source terminals. The general structure of Gate Matrix Layout is shown in Figure 2.6.

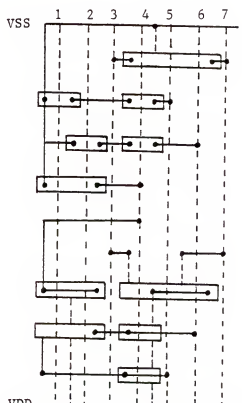
Only a couple of researchers have proposed automatic layout algorithms [23-26] based on Gate Matrix Layout method. Both of them used the same interval graph procedures proposed by Ohtsuki et al. They formulated an CMOS



(a)



(b)



(c)

Figure 2.6 The Gate Matrix Layout Structure. a) Vertical Grid Structure; b) Horizontal Grid Structure; c) Realization of a Circuit on Gate Matrix.

transistor circuit with the same netlist structure as shown in Figure 2-1. Unfortunately, since the gate matrix structure is naturally different from the Weinberger's array structure, the same netlist structure can not be applied directly. The main problem with their approach is that they neglect the vertical wiring during the problem formulation. For a Weinberger's single-gate row structure or two-device-row structure as proposed by Uehara et al [22], no vertical wiring is needed because the vertical wiring is realized by the vertical gates' or devices' area. However, since there are more than one device rows in Gate Matrix Layout structure, the vertical wiring between two p-device rows, one p-device row and one n-device row, and the device rows and the power/ground nets should be considered. Especially as the Gate Matrix is a compact layout structure, the vertical wiring, which usually causes vertical channel congestion problem, is usually the most difficult problem to deal with.

2.3 Layout Algorithms Based on Circuit Embedding

Circuit embedding is a classical problem in graph theory. Given a circuit, if we represent each component as a vertex and each wire as an edge, then there is a corresponding graph for the circuit. A drawing of a geometric representation of a graph on any surface such that no edges intersect is called embedding. This kind of approach assumes no layout structure and is regarded as a different category from the previous two layout methodologies.

The circuit embedding method was first applied by Rose and Oldfield [27] for the embedding in single-layer circuit boards. Engl et al. [28] used a graph theoretical approach to embed bipolar analog circuits into integrated circuits. Van Lier and Otten [29] used a similar approach to solve the problem of embedding bipolar analog circuits in monolithic technology. Ng and Johnsson [30]

used the same approach to develop automatic layout algorithms for MOS circuit. In his approach, he modeled a circuit as a special circuit graph. He then used a path finding algorithm, developed by Hopcroft and Tarjan [37], to find the hierarchy of paths which record the information of the whole circuit. A couple of constraint graphs are then derived from the path tree which records the topological constraints of the paths. The circuit embedding topology can be generated by simultaneous assignment or gradual assignment of the paths which satisfy the topological constraints.

2.4 Discussion

In this survey, we note that most of the approaches treated cell design as a problem which was separated from chip design. Each algorithm emphasized only minimizing the individual cell area and offered limited study of the impact of the cells on the chip design. In practical VLSI design, however, cell configuration is usually more important than cell area in global optimization of chip area. Optimization of the cell design needs to consider the whole chip design environment.

Another problem with the previous approaches is that most of the algorithms used a simple graph model to represent the layout structure. Since the modeling is usually oversimplified and sometimes inaccurate, the implementation result can be far different from the theoretical result. Consequently, even if the theoretical result proves to be good, the physical layout may be poor.

In retrospect, if we are to design an automatic cell generation system, it is important that we make note of the following points:

(1) Cell layout generation is not a stand-alone problem. Optimization of the cell design is meaningless unless it can improve the performance

of the chip.

(2) The constraints of the layout domain have to be relaxed somewhat to allow certain flexibility in cell configurations.

(3) A delicate modeling technique is needed to accurately transcend the transformation between the graph model and symbolic layout structure, and the symbolic layout structure and physical layout structure.

CHAPTER III
CELL LAYOUT GENERATION PROBLEM OVERVIEW

The general cell layout method, as mentioned previously, has become the most popular approach for efficient VLSI layout. Previous methods applicable to this layout approach include min-cut/slicing [8,38], constructive [39], analytic [40-42], and simulated annealing [43,44]. In general, the general cell layout approach can fully take advantage of the design flexibility in cell configurations and is expected to achieve higher chip density and chip performance than the polycell layout approach. However, in most of the current systems, the chip floor planning and cell generation processes are considered separately. The lack of real-time interaction between these two processes has severely degraded the performance of general cell layout systems. Sizable redundant chip areas are created due to the shape mismatch between the allotted chip subarea and generated cell of each subcircuit. Laborious cell redesign work is required to reduce such redundant chip areas.

In consideration of the practical design environment, an integrated, flexible-cell layout system has been developed. An automatic cell generation system is introduced into the general cell VLSI layout environment to provide real-time interaction with the automatic chip floor planner. The redundant chip area caused by cell shape mismatch will be greatly reduced through the real time, iterative improvement.

Four sections are included in this chapter. In the first section, the structure of the integrated, flexible-cell layout system is described. The function of the proposed automatic, flexible cell generation system is also discussed. In the

second section, a symbolic cell layout design methodology, Gate Matrix Layout method, is described. A cell structure model, which links the symbolic layout environment with the physical layout environment, is also presented. In the third section, the symbolic cell layout problems are formulated. The layout problem is divided into several subproblems. Finally, the last section presents the system structure of the cell layout generation process which solves the formulated cell layout problems.

3.1 Overview of the Integrated Flexible-Cell Layout System

During the initial synthesis stages of IC design, designers attempt to partition chip area and choose attributes, e.g., areas, aspect ratios, and I/O pin locations, for each partitioned chip subarea. However, it is difficult for designers to comprehend the complicated relationship between initial design decisions and the final global assignment of modules. Moreover, the consequences of the designer's decisions may not become apparent until after a significant amount of the IC design process has been performed. Consequently, it is often the case that a great deal of iteration is required between the chip floor planning and the cell layout design tasks in order to produce a satisfactory design. In order to achieve such kind of real-time interaction between chip floor planning and cell layout design, an automatic cell generation system is introduced in the general cell layout system. Figure 3.1 shows the functional flow of the integrated flexible-cell layout system. An automatic chip floor planner partitions the chip area into chip subareas according to the input cell attributes such as cell area, cell aspect ratio, and inter-cell connectivity. The cell generation system then produces custom-fit cells from a transistor connectivity description and a description of the environment in which

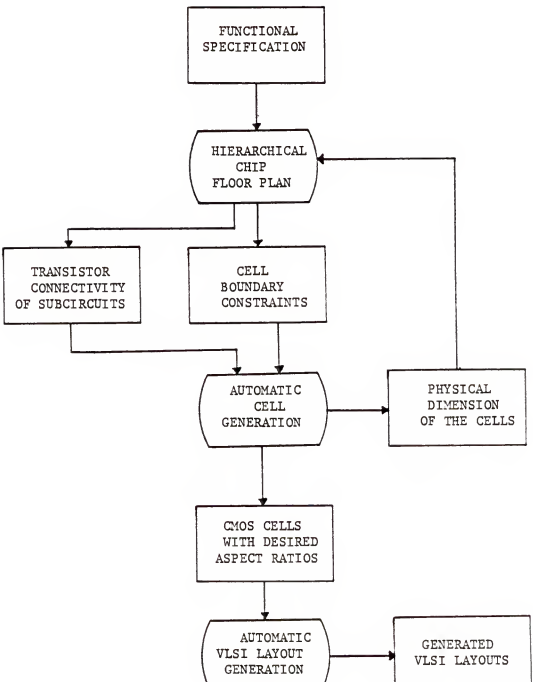


Figure 3.1 Functional Flow of the Integrated Flexible-cell Layout System.

the cell will reside.

The cell generation system requires two inputs as shown in Figure 3.2. First, the circuit must be defined in terms of transistor connectivity. Second, the cell generation system requires information about the environment in which the cell will reside. The boundary constraints, which include block dimensions, aspect ratio, and external signal placement, are given by the chip floor planner with an emphasis to optimize the chip performance. The cell generation system then synthesizes a CMOS symbolic cell layout. A cell compactor is used to convert the symbolic description to a mask level description as shown in Figure 3.3. Because the symbolic layout process is isolated from the design rules, it is assured of technology-independence. However, since symbolic layout and physical layout are operated in different layout domains, a cell structure model is used to guide the transition between different layout domains (see Figure 3.4). Figure 3.6 displays the transistor connectivity description of a compare circuit which is shown in Figure 3.5. The textual description is based on the language developed by GE [45] and EDIF group [46]. Next, the environment in which the compare cell will reside is defined as in Figure 3.7 and is depicted in Figure 3.8. Given the environment and the transistor connectivity, the cell generation system will create a symbolic layout for the compare cell (see Figure 3.9). The symbolic layout will be inserted into the floor plan which expects to minimize the redundant chip area.

3.2 The Structure Model

As described in Chapter II, the success of an automatic cell layout algorithm depends on whether it can exploit the design search space with underlying modularity and geometric simplicity. In other words, a structured design

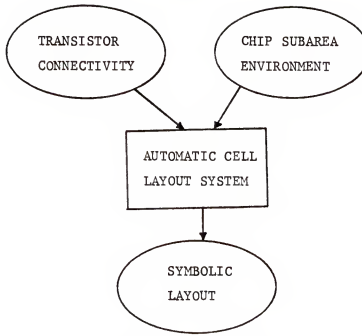


Figure 3.2 The Symbolic Layout Process.

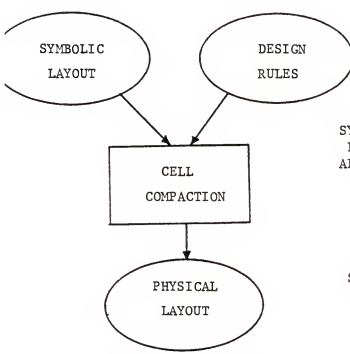


Figure 3.3 The Cell Compaction Process.

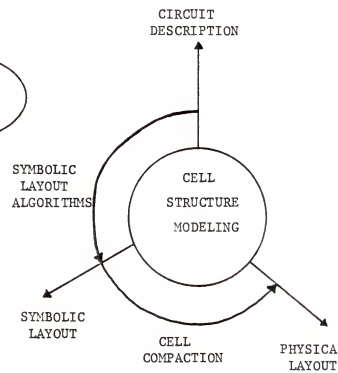
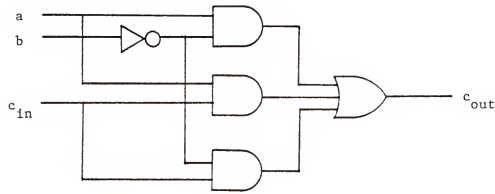
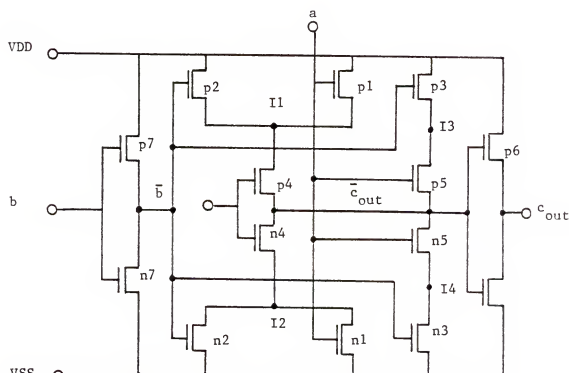


Figure 3.4 Transition among Different Circuit Domains.



(a)



(b)

Figure 3.5 A Compare Circuit. a) Gate-level Circuit; b) Transistor Circuit.

```
(cell compare
  (structure
    (transistor
      (signal
        (a ((gate p1) (gate p5) (gate n1) (gate n5)))
        (b ((gate p7) (gate n7)))
        (c_in ((gate p4) (gate n4)))
        (b_bar ((source p7) (drain n7) (gate p2)
                (gate p3) (gate n2) (gate n3)))
        (c_out_bar ((source p4) (source p5) (gate p6)
                    (drain n4) (drain n5) (gate n6)))
        (I1 ((source p1) (source p2) (drain p4)))
        (I2 ((drain n1) (drain n2) (source n4)))
        (I3 ((source p3) (drain p5)))
        (I4 ((drain n3) (source n5)))
        (c_out ((source p6) (drain n6)))
        (vdd ((drain p1) (drain p2) (drain p3)
              (drain p6) (drain p7)))
        (vss ((source n1) (source n2) (source n3)
              (source n6) (source n7)))
      (instance
        (p_type (p1 p2 p3 p4 p5 p6 p7))
        (n_type (n1 n2 n3 n4 n5 n6 n7))))))
```

Figure 3.6 Transistor Circuit Description of Figure 3.5(b).

```
CRITICAL PATH OF THE CHIP FLOOR PLAN:
   HORIZONTAL = 260
   VERTICAL = 240

LONGEST PATH OF THE CHIP FLOOR PLAN
WHICH PASSES THROUGH THE CELL:
   HORIZONTAL = 260
   VERTICAL = 210

ORIGINAL CELL DIMENSION:
   WIDTH = 80
   HEIGHT = 30

SUGGESTED CELL DIMENSION:
   WIDTH = 40-60
   HEIGHT = 50-70

SUGGESTED I/O LOCATIONS:
   TOP --- VDD
   BOTTOM --- VSS
   LEFT --- b
   RIGHT --- C_out
```

Figure 3.7 Cell Boundary Information Issued by Chip Floor Planner.

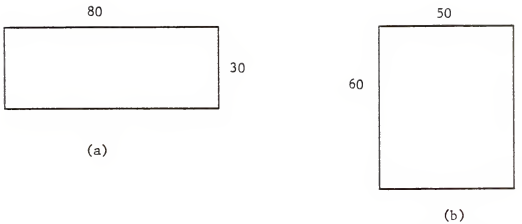


Figure 3.8 Cell Configuration Suggested by the Cell Boundary Information. a) Original Cell Configuration; b) New Cell Configuration.

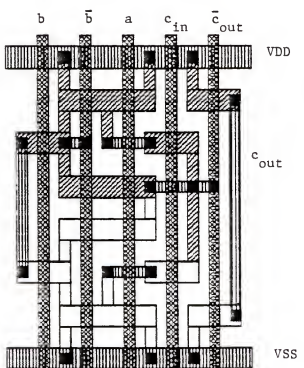


Figure 3.9 Symbolic Layout of the Compare Cell.

methodology with predictable layout environment is essential for the cell layout design automation. However, the layout structure should be flexible enough to ensure the design flexibility in cell configurations. Among the well known cell layout methods, we chose Gate Matrix Layout structures over other layout styles for the following reasons:

(1) The cell structure, with power net and ground net running horizontally and polysilicon line running vertically, makes both the inter-cell abutment and the cell stacking easy to be implemented. Consequently, the chip floor planning is much easier with those cells.

(2) The underlying grid structure provides a discrete, predictable design space for both device placement and routing. The development of layout algorithms is much easier under the layout environment.

(3) The aspect ratio of the cells can be adjusted by relaxing a constraint assumed by Gate Matrix Layout. Each gate signal can be assigned to more than one device column. Therefore, the aspect ratio of a cell can be adjusted by changing the number of device columns.

3.2.1 The Grid Structure of Gate Matrix Layout

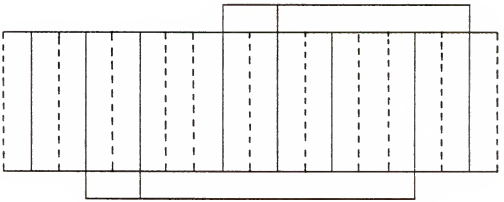
The Gate Matrix Layout method [9] provides a discrete layout structure for the determination of the topology of the layout primitives. The layout primitives include the basic elements like transistors (devices), wires, and contacts. Each element is represented as a special symbol. The symbols are then placed in an equally spaced grid matrix with an emphasis to optimize certain objective functions. The grid structure consists of two sets of grids. Vertical grids are composed of an array of polysilicon grids and diffusion grids as shown in Figure 2-6(a). In the original Gate Matrix structure proposed in Lopez and Law [9]. each

gate signal can be assigned to only one polysilicon grid and only one diffusion grid is allowed between each two adjacent polysilicon grids. Those layout rules severely limit the flexibility of cell configurations. In this dissertation, we modify the layout structure such that each gate signal can be implemented as a set of interconnected polysilicon grids. More than one diffusion grid can be inserted into each pair of adjacent polysilicon grids. The modified vertical grid structure is shown in Figure 3.10(a). With such modification, more cell configurations can be enumerated. Moreover, more routing algorithms can be chosen with the less rigid layout structure.

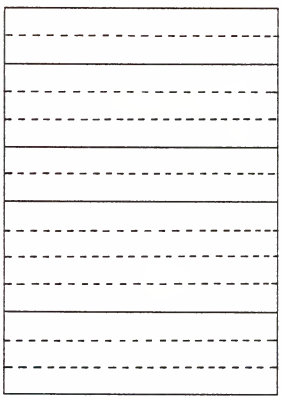
The horizontal structure of the matrix is composed of power net, ground net, device rows, and horizontal routing channels as shown in Figure 2.6(b) which is redrawn in Figure 3.10(b). The power net and ground net are running at the top and bottom of the cell respectively. Devices are placed on horizontal grids which are called device rows. Horizontal wire segments which connect the devices can be placed on either device rows or horizontal routing channels. In order to simplify the design, horizontal wires are usually realized by metal while the vertical wires are realized by diffusion line.

3.2.2 The Cell Structure.

With the grid structure described in section 3.2.1, the physical cell dimension can be represented as a cell structure model as shown in Figure 3.11. In vertical dimension, the cell consists of device rows and horizontal routing tracks. Let the height of each device row be h_r and the height of each horizontal routing track be h_t, then the cell height is given by



(a)



(b)

Figure 3.10 Revised Gate Matrix Layout Structure. a) Vertical Grid Structure; b) Horizontal Grid Structure.

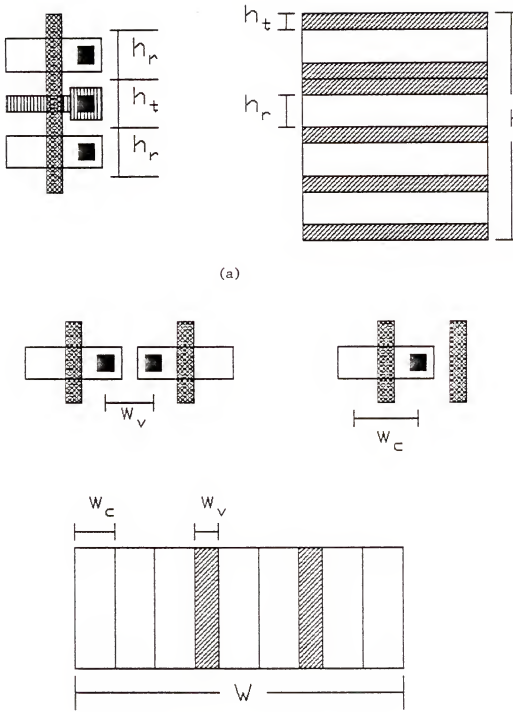


Figure 3.11 The Cell Structure Model. a) Vertical Cell Structure; b) Horizontal Cell Structure.

$$H = n_r h_r + n_t h_t \tag{3-1}$$

where n_r is the number of device rows and n_t is the number of horizontal routing tracks. In horizontal dimension, the cell consists of device columns and inserted routing tracks. Let the width of a device column be w_c and the width of an inserted vertical routing track be w_v , then the cell width is given by

$$W = n_c w_c + n_v w_v \tag{3-2}$$

where n_c is the number of device columns and n_v is the number of vertical routing tracks.

3.3 Problem Formulation

One of the most difficult features of the cell generation problem is to generate the cells with desired configurations. Since the cell configurations are usually bounded by their layout structures, it is essential to determine an appropriate grid structure before the symbolic layout process may begin. Therefore, the cell generation problem is divided into two major subproblems: cell boundary analysis and symbolic layout. The process of cell boundary analysis (see Figure 3.12) analyzes the circuit information and the boundary information to determine a grid structure which will lead to a desired cell configuration. The symbolic layout process (see Figure 3.13) then uses the grid structure to realize the circuit connectivity with an optimized object function.

To define these subproblems more precisely, we introduce some notations and

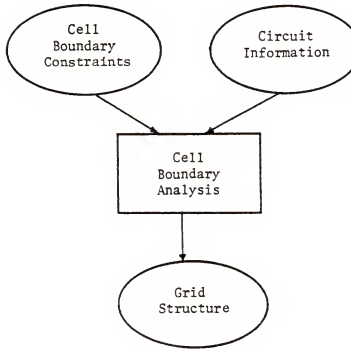


Figure 3.12 The Cell Boundary Analysis Process.

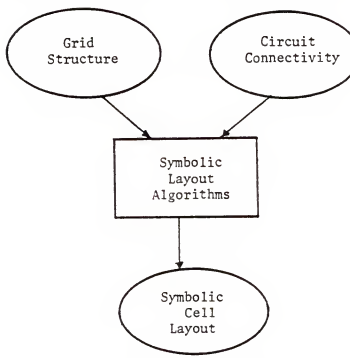


Figure 3.13 The Symbolic Layout Process.

definitions. Let $T = \{ t_i \mid t_i$ is a transistor$\}$ be a set of transistors. Let $S = \{ S_i \mid S_i$ is a net$\}$ be a set of nets which describes the circuit connectivity. Each net S_i is described as a set of interconnected device terminals, $S_i = \{p_{i1}, p_{i2}, \ldots, p_{ik}\}$. Let C be a set of columns and R be a set of rows. Let B be the set of boundary constraints given by the chip floor planner. Now the cell boundary analysis problem is described as

[Grid Assignment Problem]: Given a set of transistors T, a set of nets S which describes the connectivity of T, and a set of boundary constraints B, find an assignment function $f_1 : (T,S,B) \Longrightarrow (C,R)$ such that (1) all the transistors and connectivity can be realized within the grid structure (C,R) (2) The grid structure (C,R) will lead to a desired cell configuration.

Now we formulate the symbolic layout problem. As shown in Figure 3.14, a circuit is composed of a set of interconnected transistors. Each transistor consists of three terminals: drain, gate, and source. Those transistor terminals and their connectivity are to be embedded in a grid structure which is determined in the grid assignment process. Figure 3.15 shows two grid structures with different cell configurations. Each grid structure contains P grid locations which can be used for the placement of transistor terminals. Obviously, the permutation number PN for those transistor terminals within the grid locations is very huge. If we assume that the number of transistors of a circuit is N, the permutation number for all 3N transistor terminals among P grid locations $(P \geq 3N)$ is

$$PN = C(P,3N)(3N)!$$

The computation complexity is very high even with small number of P and N. Fortunately, with the help of the layout constraints from the Gate Matrix structure, the computation complexity can be reduced tremendously. Figure

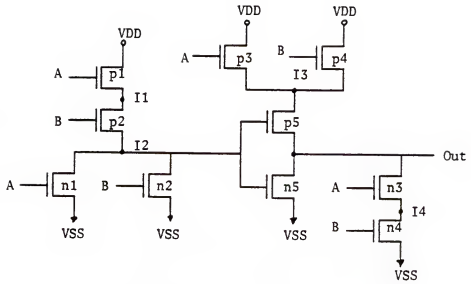
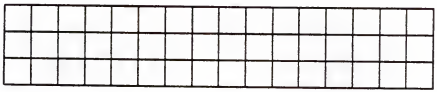
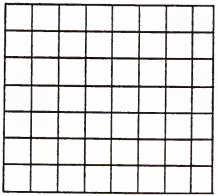


Figure 3.14 Transistor Circuit of an Exclusive OR Logic Gate.



(a)



(b)

Figure 3.15 Two Possible Grid Structures for the Layout of the Circuit in Figure 3.14. a) Wide-Shape Structure with 17 Vertical Grids and 4 Horizontal Grids; b) Long-Shape Structure with 9 Vertical Grids and 8 Horizontal Grids.

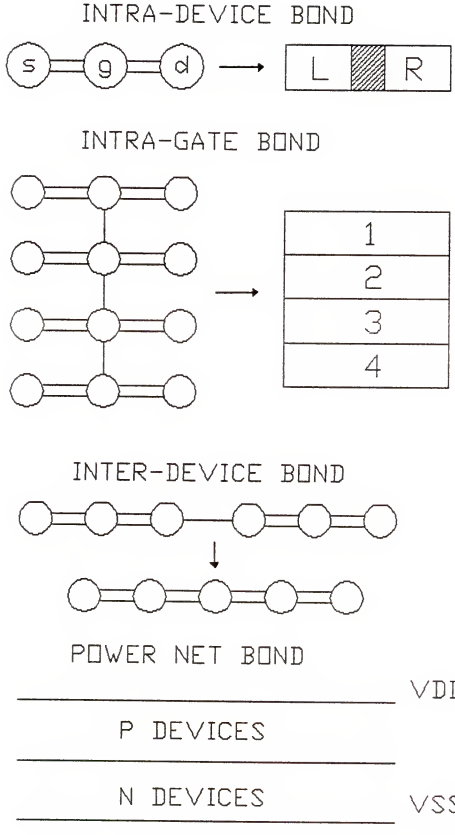


Figure 3.16 Four kinds of Constraints for Gate Matrix Structure.

3.16 shows four kinds of constraints of the layout structure. Category I is called "intra-device bond" where the two terminals of each device have to locate at adjacent horizontal locations. However, these two terminals are allowed to switch positions. Category II is called "intra-gate bond" where all the devices which are driven by the same polysilicon gate have to stay at the same column. The devices, however, are allowed to permute among the grid locations in that column. Category III is called "inter-device bond" where two adjacent devices are allowed to share a common terminal location if those two terminals are common electrically. However, those two terminals should have different grid locations if they are not common electrically. Category IV is called "power net bond" where p devices and n devices are usually placed at different regions which are closer to the power net and ground net respectively. By using all those geometric constraints, the layout problem can be decomposed into several assignment functions. Let $G' = \{ g'_i \mid g'_i$ is a gate column$\}$ be a set of gate columns. Each gate column g'_i is composed of a set of devices, $g'_i = \{t_{i1}, t_{i2}, \ldots, t_{ik}\}$. Let $O = \{$east,west$\}$ be a set of orientations. Now the device placement subproblem can be described as

[Placement Problem]: Given a set of gate signals G, a set of transistors T, a set of device columns C, a set of device rows R, and a set of orientations O, find the assignment functions $f_2 : G \dashrightarrow C$, $f_3 : T \dashrightarrow O$, $f_4 : T \dashrightarrow R$ such that

(1) The assignment functions results in a device topology which is completely routable.

(2) The total wiring length for the device topology is minimized.

Now we define the routing problem. Let $S' = \{ S'_i \mid S'_i$ is a net$\}$ be a set of nets which are described by grid points. Each net $S'_i = \{ p'_{i1}, p'_{i2}, \ldots, p'_{ik} \}$ where $p'_{ik} = (r_{ik}, c_{ik})$ represents a grid point with coordinates (r_{ik}, c_{ik}). Let S"

$= \{S''_1, S''_2, \ldots, S''_k\}$ be a set of nets which are described by grid wire segments. Each net $S''_i = \{W''_{i1}, W''_{i2}, \ldots, W''_{ik-1}\}$ where $W''_{ik} = [(r_{ik1}, c_{ik1}), (r_{ik2}, c_{ik2})]$ is a wire segment described by two grid points. The routing problem is described as

[Routing Problem] Given a 3-tuple assignment function (f_2, f_3, f_4) and a functional description of a net S, find the assignment functions $f_5 : S \dashrightarrow S'$ and $f_6 : S' \dashrightarrow S''$ such that the total wiring length is minimized.

3.4 Automatic Cell Layout Generation Process

The implementation of the symbolic topological layout system is described in this section. Figure 3.17 shows the function flow of the system which is composed of four major tasks: a circuit analyzer, a placer, a router, and a layout improvement mechanism. The circuit analyzer is used to analyze the lexical structure of the circuit description and the cell boundary constraints of the environment. It then determines the grid structure and generates a set of data structures for topological layout process. The placer is divided into several subtasks. Each subtask is designed to individually solve a subproblem described in section 3.3. The subtasks are gate column order assignment. device orientation assignment, device row order assignment. Gate column order assignment is the task which determines the horizontal grid location of each gate column. Device orientation assignment is to determine the relative grid locations of the two terminals of each device. Device row order assignment is to determine the vertical grid location of each device. The router of the system contains three subtasks: terminal labeling, wire-list generation, and routing. Terminal labeling is to label the net number of each device terminal on its grid location. The generated data structure is then used by the other two subtasks

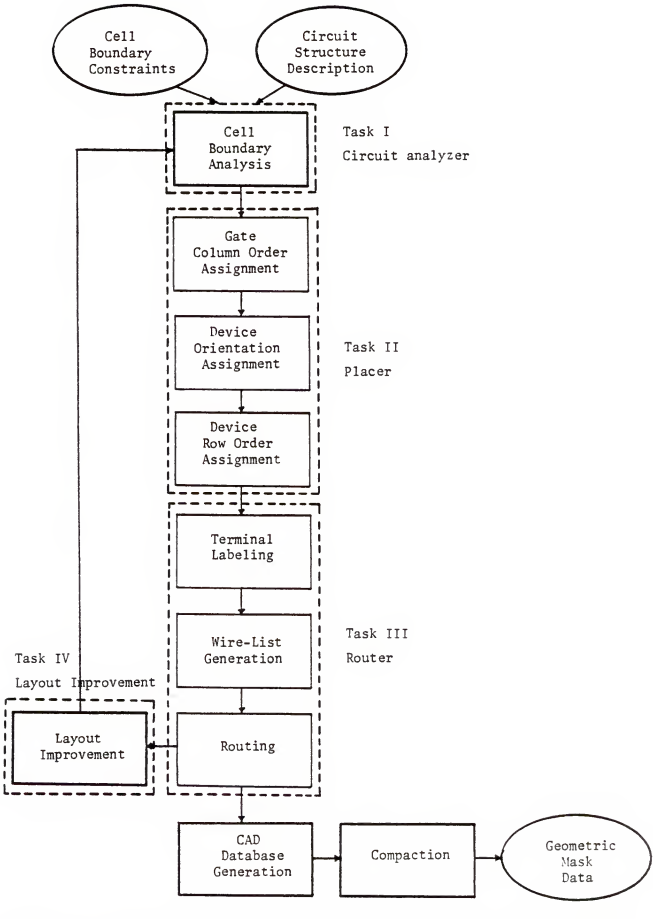


Figure 3.17 Functional Flow of the Automatic Cell Layout Generation System.

for fast searching. Wire segment generation is to determine the shortest path which realizes the circuit connectivity. Routing is to determine the grid locations for the shortest path. A layout improvement task is also introduced in the system to improve the layout result if it is found unsatisfactory in cell area or cell configuration. In this symbolic layout system, a final compaction is needed to convert symbolic layout into physical mask data. The algorithms can be seen in various studies [10,11,13] which will not be discussed in this dissertation.

CHAPTER IV
ANALYSIS OF CELL BOUNDARY CONSTRAINTS

As described in section 3.3, the cell generation problem is divided into two major subproblems: cell boundary analysis and symbolic layout. The process of cell boundary analysis is to determine a grid structure based upon the analysis of cell boundary conditions and circuit connectivity. There are two reasons for determining a grid structure before the symbolic layout process. First, a grid structure provides a layout environment for discrete, deterministic analysis. Computation is based on the discrete structure. Second, cells laid out on a grid structure with different row/column combinations will have different cell configurations. By controlling the row and column numbers of the grid structure, a cell with desired configuration will be generated. In this chapter, the process of determining the discrete grid structure is presented. The process uses the cell structure model described in Chapter III to determine the layout grid structure from the analysis of physical boundary constraints.

Three sections are included in the chapter. In the first section, the relationship between the cell configuration and chip floor plan is analyzed. The problem to determine the cell configuration is formulated as a quadratic programming problem. Objective function for this problem is derived based upon the analysis of the chip floor plan. The second section describes the equality and inequality constraints for this problem which are derived from the analysis of the grid structure and circuit structure. In the third section, procedures of solving the quadratic programming problem are presented. Examples are given for the illustration of the procedures.

4.1 Analysis of Chip Floor Plan

Chip floor planning is the process of dividing the available chip area into a number of rectangular subareas that best satisfy the area, connection, and performance constraints imposed by the design specification. An example of a chip floor plan is shown in Figure 4.1. Each chip subarea in the figure is designated for the placement of a cell of the circuit. Since a chip floor plan is composed of cells with irregular shapes and sizes, shape mismatch between neighboring cells usually produces sizable dead space in chip area. The shape mismatch between a cell and its approtioned chip subarea also causes considerable dead space (see Figure 4.2). Consequently, iterative improvements of a chip floor plan are needed. The purpose of this section is to develop a sysmatic method to analyze a chip floor plan and assign new chip subareas to the cells such that the dead space in the chip area is reduced. To facilitate the analysis process, we define two special directed graphs and use them to represent the topological information of the cells and chip subareas of a chip floor plan. Let $G_x = \{ V_x, E_x \}$, be a horizontal constraint graph [10] and $G_y = \{ V_y, E_y \}$ be a vertical constraint graph. Each constraint graph represents the relative placement of the modules (cells or chip subareas) along one dimension. Modules are shown as vertices in the graph. Each edge e_x (e_y) in the horizontal (vertical) constraint graph is represented as a pair of ordered vertices (v_i, v_t). The vertex, which edge e_x is incident out of, is called the initial vertex of e_x. The vertex, which e_x is incident into is called the terminal vertex. For a horizontal constraint graph, an edge (v_i, v_t) indicates that two modules are horizontally abutted and that v_t lies to the right of v_i. An edge (v_i, v_t) in a vertical constraint graph indicates that two modules are abutted vertically and that v_t lies to the bottom of v_i. Each vertex v_x (v_y) of the horizontal (vertical) graph is assigned a weight which is equal to

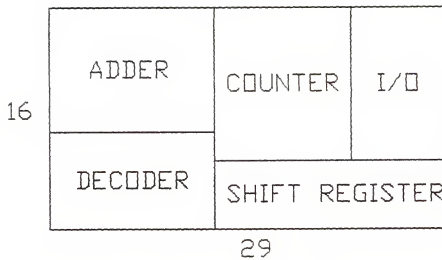


Figure 4.1 A Sample Chip Floor Plan.

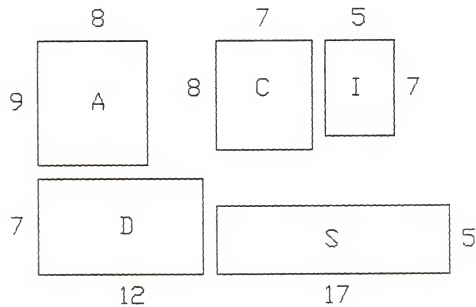


Figure 4.2 Physical Dimensions of the Cells in Figure 4.1.

the width (height) of its represented module. Let w_x (v_x) be the weight of a vertex v_x in horizontal constraint graph and w_y (v_y) be the weight of a vertex v_y in vertical constraint graph. Let W_A and H_A be the width and height of module A which is represented as vertex v_i and v_j in the horizontal and vertical constraint graphs respectively; then

$$w_x(v_i)=W_A \tag{4-1}$$

$$w_y(v_j)=H_A \tag{4-2}$$

The weight of an edge is equal to the weight of its initial vertex. Let $x_w(A)$ denote the position variable of the western boundary of the chip subarea assigned for module A. Functions $x_e(A)$, $y_n(A)$, and $y_s(A)$ are similarly defined as the eastern boundary, northern boundary, and southern boundary of module A. For a horizontal graph, a directed edge from vertex A to B indicates the following conditions:

$$x_e(A) = x_w(B) \tag{4-3}$$

$$y_s(A) < y_n(B) \tag{4-4}$$

$$y_s(B) < y_n(A) \tag{4-5}$$

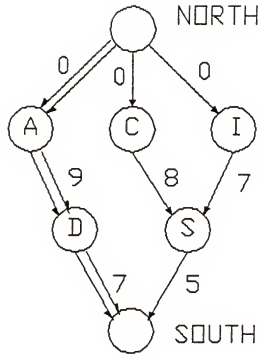
It means that the chip subareas for functional modules A and B are horizontally abutted and that module A is located at west side of module B. Similar conditions can be derived for a directed edge in the vertical constraint graph:

$$y_s(A) = y_n(B) \tag{4-6}$$

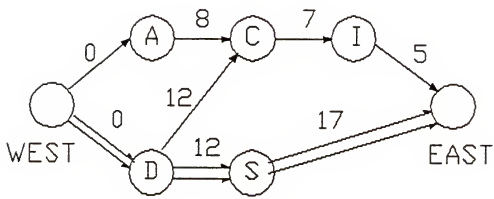
$$x_w(A) < x_e(B) \tag{4-7}$$

$$x_w(B) < x_e(A) \tag{4-8}$$

It is obvious that the width (height) of the chip area CW (CH) is equal to the length of the longest directed path in the horizontal (vertical) constraint graph. The longest directed path of the horizontal constraint graph is called a horizontal critical path (HCP). The same definition is applied for vertical critical path (VCP). The vertices and edges in a HCP (VCP) are called horizontal (vertical) critical events and the horizontal (vertical) critical activities. Figure 4.3 depicts the two constraint graphs for the chip floor plan in Figure 4.1. Critical paths in both of the graphs are shown as double lines. VCP contains critical events of A and D. HCP contains critical events of D and S. The length of VCP is 16 which is equal to the height of the chip area in Figure 4.1. The length of HCP is 29 which is equal to the chip width in Figure 4.1. Let L(P) denote the length of any path P. Then the chip dimension CH and CW can be represented as the summation of all the cell dimensions along the respective critical path



(a)



(b)

Figure 4.3 Constraint Graphs for the Chip Floor Plan of Figure 4.1 and Figure 4.2. a) Vertical Constraint Graph; b) Horizontal Constraint Graph. The Weight of Each Edge Represents the Cell Dimension of the Initial Vertex of that Edge.

$$CH = L(HCP) = \sum_{v_i \epsilon HCP} w_x(v_i) \tag{4-9}$$

$$CW = L(VCP) = \sum_{v_j \epsilon VCP} w_y(v_j) \tag{4-10}$$

4.1.1 Optimization of Chip Floor Plan

The objective function of the chip floor planning process is to optimize the total chip area. In other words, it is to minimize HCP and VCP along each dimension. There is no known efficient technique for producing an optimal solution to this problem. Accordingly, we developed a heuristic algorithm to iteratively improve the chip floor plan. The algorithm is to find a module in the HCP or VCP (but not both) and to reconfigure the chosen module such that the length of one of the critical paths is reduced while the length of the other one remains the same. Let HSL (VSL) denote the second longest path in the horizontal (vertical) constraint graph. Suppose that we choose a module m which is a critical event of HCP and is not a critical event of VCP. Then the length of HCP is equal to the sum of the width of module m and the width of the other modules in the HCP:

$$L(HCP) = W_m + L(HCP - \{m\}) \tag{4-11}$$

By reducing the width of module m, the length of the horizontal critical path L(HCP) will be reduced. However, there is an upper bound for the reduction of L(HCP):

$$\Delta W = L(HCP) - L(HSL) \tag{4-12}$$

It means that whenever L(HCP) is reduced more than the amount of ΔW, it is shorter than another path HSL. In that case, HSL becomes the new horizontal critical path. Therefore any further size reduction of the cells along the original HCP will not help in reducing the chip dimension.

Now we consider the other dimension of the chip area. Since the decrease of one cell dimension causes the increase of the other dimension, we have to make sure that the increase of the cell height, which is caused by cell reconfiguration, does not increase the length of VCP. Because we have assumed that module $m \epsilon VCP$, the upper bound which is allowed for the increase of the cell height while the length of VCP is not increased is

$$\Delta H = L(VCP) - L(VLP_m) \tag{4-13}$$

where VLP_m represents the longest path in vertical constraint graph which passes through module m.

By adding ΔH to the original cell height H_m and reducing ΔW from the original cell width W_m, the new cell boundary becomes

$$W_b = W_m - \Delta W \tag{4-14}$$

$$H_b = H_m + \Delta H \tag{4-15}$$

Similar equations will be derived if we choose a module which is a critical event of VCP and is not a critical event of HCP

$$\Delta H = L(VCP) - L(VSL) \tag{4-16}$$

$$\Delta W = L(HCP) - L(HLP_m) \tag{4-17}$$

$$W_b = W_m + \Delta W \tag{4-18}$$

$$H_b = H_m - \Delta H \tag{4-19}$$

The new cell boundary represents the best cell configuration suggested for the the chosen module to be reconfigured. If the module can be reconfigured to a new configuration which fits into the cell boundary, the chip floor plan will be improved. To illustrate the chip floor plan improvement process more clearly, we follow the same example in Figure 4.1. From Figure 4.3, we know that module A is a critical event of VCP but not a critical event of HCP. By reconfiguring module A into a different shape as shown in Figure 4.4, the original chip floor plan will be reduced to a size of 29x13 (see Figure 4.5).

4.1.2 Objective Function for Cell Reconfiguration

After the new cell boundary is calculated, one of the major objectives for

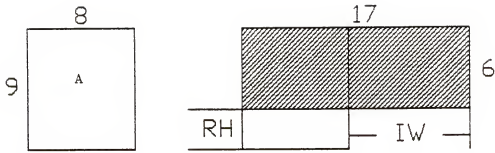


Figure 4.4 Reconfiguration of the Adder Cell with a Different Shape.

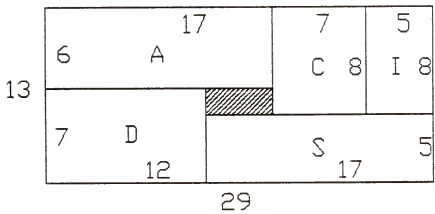


Figure 4.5 The Chip Area Is Reduced after the Cell Reconfiguration.

the cell reconfiguration is to determine a new cell configuration which best fits the new cell boundary. The configuration of the reconfigured cell may still not fit perfectly with the new designated subarea. However, the main purpose of this cell reconfiguration is to try to reduce the redundant chip area as much as possible. Figure 4.6 shows several situations which cause redundant chip area. In case 1, both dimensions of the cells are longer than those of the cell boundaries. In case 2, the height of the cell exceeds the cell boundary while the cell width stays within the boundary. In case 3, the width of the cell exceeds the cell boundary while the cell height stays within the boundary. In case 4, neither of the cell dimension exceeds the boundary. Let W and H denote the width and height of the cell to be generated. Then the objective function of the cell reconfiguration is to minimize the redundant cell area which exceeds the designated cell boundary. If several cell configurations cause the same amount of redundant cell area, the one with the minimum cell area will be chosen. Consequently, the objective function can be represented as

minimize

$$y = RA = \alpha W(H - H_b) + \beta H(W - W_b) - \alpha\beta(H - H_b)(W - W_b) \tag{4-20}$$

where $\alpha = 1$ if $H > H_b$; $\alpha = 0$ if $H \leq H_b$

(4-21)

$\beta = 1$ if $W > W_b$; $\beta = 0$ if $H \leq H_b$

(4-22)

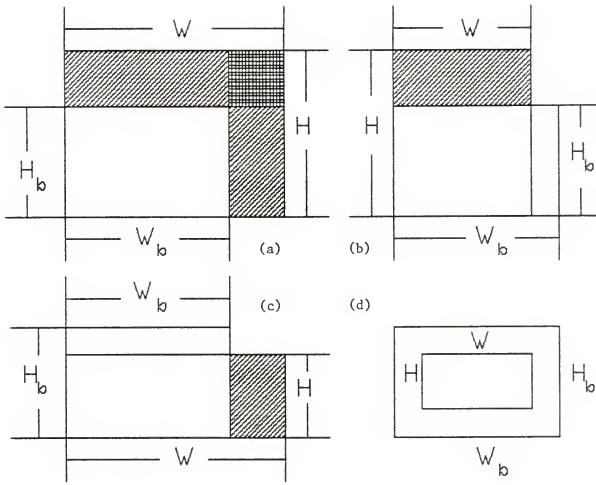


Figure 4.6 Four kinds of Cell Shape Mismatch. a) Both of the Cell Height and Cell Width exceed the Cell Boundary; b) Only the Cell Height Exceeds the Boundary; c) Only the Cell Width Exceeds the Boundary; d) Neither of the Cell Dimensions Exceed the Boundary.

4.2 Geometric Constraints for the Cell Configuration

The Quadratic Programming problem formulated above has several geometric constraints which limit the domain of its feasible solutions. The geometric constraints come from two main sources: layout guidelines and Gate Matrix layout structure. In this section, all those geometric constraints are analyzed and formulated as equality or inequality constraint equations for the quadratic programming problem.

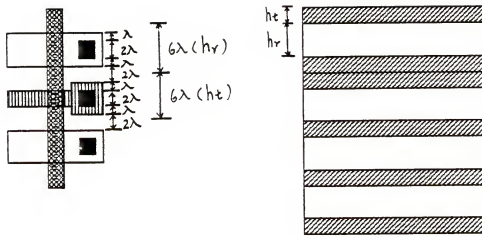
4.2.1 Constraints by Discrete Grid Structure

The boundary constraints from the chip floor plan are usually described in terms of physical dimensions. However, since the layout is implemented on a discrete grid structure, the selections of the cell configurations are limited by the number of possible combinations of integer grid column and row numbers. As discussed in Chapter II, the physical cell dimensions are represented as equations of grid numbers:

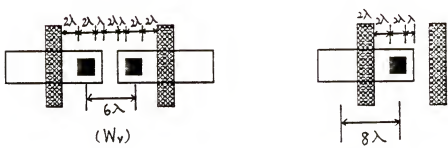
$$W = n_c w_c + n_v w_v \tag{4-23}$$

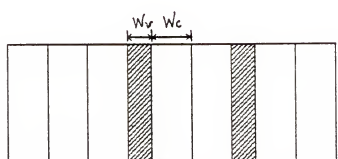
$$H = n_r h_r + n_t h_t \tag{4-24}$$

Among the equations, w_c, w_v, h_r, h_t are technology dependent which can be estimated with given design rule information. Figure 4.7 shows the estimation of the physical dimension of the Gate Matrix structure based upon the CMOS design rules described in Weste and Eshraghian [36]. Substituting the physical



(a)





(b)

Figure 4.7 Estimation of the Physical Cell Dimensions. a) Vertical Dimension; b) Horizontal Dimension.

dimensions into equations 4-20 and 4-21 yields

$$W = 8n_c + 6n_v \tag{4-25}$$

$$H = 6(n_r + n_t) \tag{4-26}$$

4.2.2 The Number of Grid Columns

In the original structure of Gate Matrix layout method, each gate signal in a circuit is assigned a grid column in the cell design. Mathematically, the number of grid columns will be equal to the number of gate signals. In the relaxed Gate Matrix layout system, each gate signal may be assigned to more than one gate signal. Therefore, the number of grid columns is now a variable instead of a constant. However, the number of grid columns is still bounded in a certain range related to the circuit connectivity. Since stacking and folding of cell structure is not considered, each signal will be assigned to at least one gate column. Consequently, the minimum number of grid columns is equal to N_g which is the number of gate signals of the circuit. Due to the assumption of parallel power and ground lines located at the top and the bottom respectively, each column requires at least 2 devices. Therefore, the maximum number of grid columns is equal to $N_d/2$ where N_d denotes the total number of devices in the circuit. Let n_c denote the number of device columns needed for a circuit to be implemented on a gate matrix. Then the range of the number of device columns can be represented as

$$N_g \leq n_c \leq \frac{N_d}{2} \tag{4-27}$$

4.2.3 The Number of Device Rows

For the Gate Matrix layout structure, the minimum number of the devices located in each device column is 2. That is also the minimum required number of the device rows. The maximum number of device rows equals the maximum number of devices assigned to any of the device columns. Let N_{di} be the number of devices located on ith device column and $MAX(N_{di})$ be the maximum number of N_{di} for all i. Then the range of the number of device rows n_r for a given circuit can be represented as

$$2 \leq n_r \leq MAX(N_{di}) \tag{4-28}$$

The exact number of device rows may depend on how many device columns are implemented and how the transistors are distributed among those device columns. Normally, the number of device rows decreases as the number of device columns increases. However, if the devices are unevenly distributed, a small increase of device columns may not change the number of device rows. Let WS denote the unused grid locations caused by uneven distribution of devices. Then the number of device rows can be represented as

$$n_r = \frac{N_d + WS}{n_c} \tag{4-29}$$

where WS can be represented as the summation of the unused grid locations in each device column

$$WS = \sum_{i=1}^{n_r} [MAX(N_{di}) - N_{di}] \tag{4-30}$$

Figure 4.8 shows an example of the row-column relationship. The number of device rows decreases as the number of device columns increases. However, the relationship is obviously not linear.

4.2.4 The Number of Routing Tracks

It is very difficult to estimate the number of routing tracks before the layout process actually starts because the information is not sufficient. Therefore, the number of routing tracks can only be roughly estimated. We start from estimating the average horizontal length of a net. In general, there are two kinds of nets whose horizontal length should be estimated differently. The nets among the interconnected device terminals are usually decomposed into a set of two-point nets. Each two-point net is usually realized as two vertically intersected wires as shown in Figure 4.9. If we assume that the probability of the location of any net terminal is evenly distributed among the horizontal grid locations, then the average horizontal length of any two-point net is

$$L_h = \frac{\sum_{l=1}^{n_c-1} l(n_c - 1)}{C(n_c, 2)} = \frac{n_c + 1}{3} \tag{4-31}$$

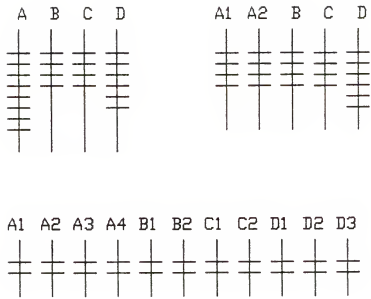


Figure 4.8 Different Grid Row and Column Combinations.

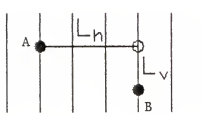


Figure 4.9 A Two-Point Net

Figure 4.10 A Multi-Point Net

However, the nets among each set of interconnected polysilicon grids can be directly implemented as a horizontal wire segment as shown in Figure 4.10. Each set of interconnected polysilicon grids usually occupies the whole horizontal routing track due to the connection to its pin location. The number of horizontal routing tracks is the sum of the routing tracks reserved for both kinds of nets.

Let S_{2-net} be the number of two-point nets in the circuit and S_{m-net} be the number of the sets of interconnected polysilicon grids. The estimated number of horizontal routing tracks is

$$n_t = \frac{S_{2-net} L_h n_r/4}{n_c} + S_{m-net} \tag{4-32}$$

where $n_r/4$ is the weight factor of the extra horizontal routing detours caused by the vertical routing obstacles. When the number of device rows increases, the total length of horizontal routing detours will also increase because there are more blocking device terminals. The number of $n_r/4$ is always rounded to its closest integer number. When n_r is smaller than 4, the weight factor is rounded to 1. For a long-shaped layout, the number of S_{m-net} is small but the number of S_{2-net} is large. Since the vertical routing distance is longer for this kind of layout structure, the average wire length for each two-point net is expected to be higher because of routing detours. For a wide-shape layout, the situation is reversed. The complete discussion of those routing situations will be covered in Chapter VI.

In Gate Matrix layout, there is only one reserved grid between each two adjacent polysilicon columns. Consequently, any two adjacent devices will be abutted. In case two devices, which do not have any common terminal, need

to be placed in adjacent locations, a grid column has to be inserted to separate those two devices. Therefore, the number of inserted vertical routing tracks is related to the number of different net segments which have to be placed at the same device row. The number n_v is increased whenever more nets have to be squeezed into fewer number of device rows

$$n_v = \frac{S_{2-net} + S_{m-net} - n_r}{2} \tag{4-33}$$

4.2.5 Determination of the Grid Structure

The constraints from both chip floor plan and circuit itself represent a set of equality and inequality constraints. Those constraints form a set of boundaries for the feasible solutions of (n_c, n_t, n_r, n_v). Combined with the objective function formulated in section 4.1.2, the complete problem for determining the grid structure can be described as

minimize

$$y = RA = \alpha W(H - H_b) + \beta H(W - W_b) - \alpha\beta(H - H_b)(W - W_b) \tag{4-34}$$

subject to

$$\alpha = 1 \text{ if } H > H_b \; ; \; \alpha = 0 \text{ if } H \leq H_b \; ; \tag{4-35}$$

$\beta = 1$ if $W > W_b$; $\beta = 0$ if $W \leq W_b$;

(4-36)

$$W = 8n_c + 6n_v$$

(4-37)

$$H = 6(n_r + n_t)$$

(4-38)

$$N_g \leq n_c \leq N_d$$

(4-39)

$$2 \leq n_r \leq MAX(N_{di})$$

(4-40)

$$n_r = \frac{N_d + WS}{n_c}$$

(4-41)

$$WS = \sum_{i=1}^{n_c} [MAX(N_{di}) - N_{di}]$$

(4-42)

$$n_t = \frac{S_{2-net} L_h n_r / 4}{n_c} + S_{m-net}$$

(4-43)

$$L_h = \frac{n_c + 1}{3} \tag{4-44}$$

$$n_v = \frac{S_{2-net} + S_{m-net} - n_r}{2} \tag{4-45}$$

4.2.6 The Enumeration Method

Because most of the equations are highly interrelated, there is no known efficient algorithms to solve the quadratic programming problem simultaneously. In this dissertation, we use an enumerative method to find the optimal solution. The objective function is calculated over all the possible grid numbers. Since all the three grid numbers n_r, n_t, and n_v are dependent on n_c, the computation complexity is in the order of $O(n_c)$. Therefore, the computation time for the enumerative approach is within acceptable range. Following are the procedures to find the solution:

(1) Enumerate all the possible device column nuber n_c by equation 4-39.

(2) Find the corresponding device row number n_r for each n_c by equations 4-40, 4-41, and 4-42.

(3) Find the corresponding number of routing track n_t for each pair of (n_c, n_r) by equations 4-43 and 4-44.

(4) Calculate W and H for each quadruple (n_c, n_r, n_t, n_v) by equations 4-37 and 4-38.

(5) Calculate the redundant area (RA) for each quadruple by substituting the result in step 4 into equations 4- 34, 4-35, and 4-36.

(6) Select the quadruple (n_c, n_r, n_t, n_v) which has the minimum RA. If

several quadruples have the same amount of RA, choose the one with the smallest cell area.

(7) Construct the grid structure based on the selected grid numbers.

Following is an example which illustrates the complete enumeration process. Figure 4.11 shows the transistor circuit of a full adder cell. Figure 4.12 is the input circuit description of the cell. The circuit analyzer of the system summarizes the circuit information as shown in Figure 4.13. Figure 4.14 depicts three different cases of cell boundary required by the chip floor planner. Based on the enumerative method, all the quadruples (n_c, n_r, n_t, n_y) are enumerated as shown in a table in Figure 4.15. All the related information W,H, and RA for each quadruple is also included in the table. In case one, a long shape cell configuration, which has the grid number of (5,8,8,1), is selected. In case two, a square-shape cell configuration with the grid number of (7,6,8,2) is selected. In case three, a wide-shape cell configuration with the grid number of (14,2,5,4) is selected. All the generated grid structures are shown in Figure 4.16.

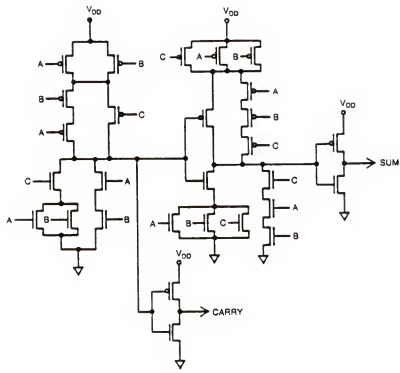


Figure 4.11 Transistor Circuit of a Full Adder Cell.

Summary of Circuit Information

Number of transistors: 28

Number of gate signals: 5

Devices distribution on gate signals: (8,8,6,4,2)

inter-net connections: 10

Number of multi-point gate nets: 2

Figure 4.13 Computer Generated Summary of the Circuit Information.

```
(cell fulladder
  (structure
    (transistor
      (signal
         (a ((gate p1) (gate p4) (gate p7) (gate p10)
             (gate n1) (gate n4) (gate n7) (gate n10)))
         (b ((gate p2) (gate p3) (gate p8) (gate p11)
             (gate n2) (gate n3) (gate n8) (gate n11)))
         (c ((gate p5) (gate p6) (gate p12)
             (gate n5) (gate n6) (gate n12)))
         (I3 ((gate p14) (gate n14) (gate p9) (gate n9)
              (source p4) (source p5) (drain n1) (drain n5)))
         (I9 ((gate p13) (gate n13) (source p12) (source p9)
              (drain n6) (drain n9)))
         (I1 ((source p1) (drain p3) (source p2) (drain p5)))
         (I2 ((source p3) (drain p4)))
         (I4 ((source n5) (drain n4) (drain n3)))
         (I5 ((source n1) (drain n2)))
         (I6 ((source p6) (source p7) (source p8)
              (drain p10) (drain p9)))
         (I7 ((source p10) (drain p11)))
         (I8 ((source p11) (drain p12)))
         (I10 ((source n9) (drain n10) (drain n11) (drain n12)))
         (I11 ((source n6) (drain n7)))
         (I12 ((source n7) (drain n8)))
         (sum ((source p13) (drain n13)))
         (carry ((source p14) (drain n14)))
         (vdd ((drain p1) (drain p2) (drain p6) (drain p7)
               (drain p8) (drain p13) (drain p14)))
         (vss ((source n2) (source n3) (source n4) (source n8)
               (source n10) (source n11) (source n12)
               (source n13) (source n14)))
      (instance
         (p_type (p1 p2 p3 p4 p5 p6 p7 p8 p9 p10
                  p11 p12 p13 p14))
         (n_type (n1 n2 n3 n4 n5 n6 n7 n8 n9 n10
                  n11 n12 n13 n14))))))
```

Figure 4.12 Computer Language Description of the Full Adder Circuit.

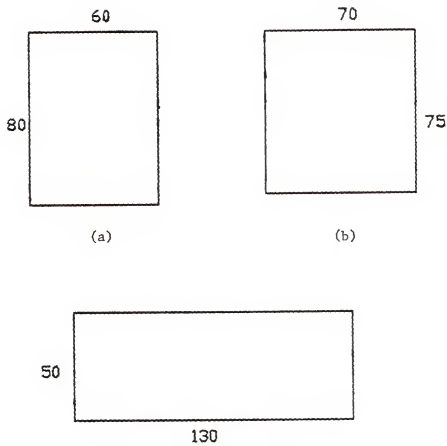
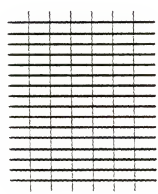


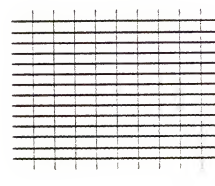
Figure 4.14 Three Different Cell Boundary Conditions. a) Case 1; b) Case 2; c) Case 3.

n_c	n_r	n_t	n_v	W	H	cell area	case 1 RA	case 2 RA	case 3 RA
5	8	8	1	46	96	4416	736*	966	2116
6	8	9	1	54	102	5508	1188	1458	2808
7	6	8	2	68	84	5712	932	612*	2312
8	4	7	3	82	66	5412	1452	792	1312
9	4	7	3	90	66	5940	1980	1320	1440
10	4	7	3	98	66	6468	2508	1848	1568
11	4	7	3	106	66	6996	3036	2376	1696
12	4	7	3	114	66	7524	3564	2904	1824
13	4	7	3	122	66	8052	4092	3432	1952
14	2	5	4	136	42	5712	3192	2772	252*

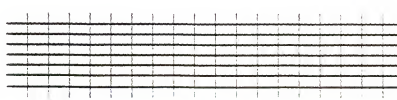
Figure 4.15 Computer Generated Result of the Enumeration Process.



(a)



(b)



(c)

Figure 4.16 Three Different Grid Structures Generated for Three Different Cell Boundary Conditions Shown in Figure 4.14. a) For Case 1; b) For Case 2; c) For Case 3.

CHAPTER V
DEVICE PLACEMENT

After the grid structure for the cell layout environment is determined, the task of the cell layout process is to determine the locations of the layout primitives on the grids. In order to reduce the computation complexity, the cell layout process in this dissertation is divided into two sequential tasks: placement and routing. Device placement is the task to determine the grid locations of all the device terminals. The routing process, which will be discussed in the next chapter, is the task to realize the circuit connectivity, with wire segments, among the device terminals.

Three kinds of grid locations are to be determined in the device placement process: horizontal grid locations of the devices, vertical grid locations of the devices, and the relative grid locations of the two device terminals (source and drain) of each device. Accordingly, the device placement problem is divided into three assignment tasks. The first section illustrates the method to determine the horizontal grid locations of the devices. This problem is formulated as a quadratic assignment problem of determining the permutation order of the device columns. The quadratic assignment problem is then solved by heuristic clustering and linear placement techniques. The second section presents the technique to determine the orientation of each device. The grid locations of all the device terminals will be determined after both the device location and device orientation are determined. Emphasis on the algorithms is to optimize the cost functions such as transistor abutment, horizontal wiring length, and vertical wiring length. Finally, the last section addresses the technique to determine

the vertical grid location of each device. The algorithms emphasize reducing the total vertical wiring length and the total device area. A vertical constraint graph based approach is used to determine the vertical grid locations of the devices which cause the minimum number of vertical wiring obstructions. Consequently, the total vertical wiring length will be minimized.

5.1 Device Column Order Assignment

In the Gate Matrix Layout structure each device is created by running a diffusion bar over a polysilicon strip. The polysilicon strip serves as the gate signal of the device. The two separated segments of the diffusion bar are used as the source and drain terminals of the device. Figure 5.1 shows the physical implementation of a device. Each gate signal is realized as either one or several interconnected vertical grids. Each vertical grid serves as a common site for a set of devices (transistors) and is called a gate column or device column. Let $G' = \{ g'_i \mid g'_i$ is a device column$\}$ be a set of device columns and each device column be composed of a set of devices, $g'_i = \{ t'_{i1}, t'_{i2}, \ldots, t'_{ik} \}$. Figure 5.2 shows the gate column and device structure. We can see from the figure that all the devices which are located on the same gate column should have the same horizontal grid location as that gate column. In other words, the horizontal grid locations of the devices will be decided once the grid locations of their gate columns are determined. Consequently, the problem of determining the horizontal grid locations of the devices can be divided into two subproblems. The first one is to convert the set of gate signals G into a number of gate columns (device columns) G'. The transistors which belong to the same original gate signal g_i will also be partitioned and assigned to several different device columns $g'_j, g'_k, \ldots, g'_l$, if the original gate signal is realized as several device columns.

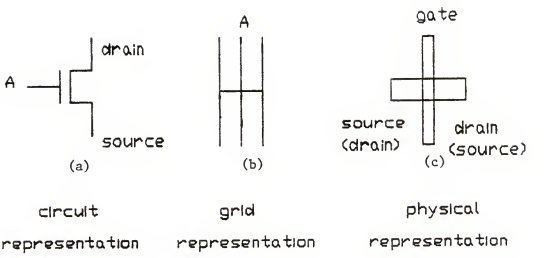


Figure 5.1 Three Different Representations of a Device. a) Circuit Representation; b) Grid Representation; c) Physical Representation.

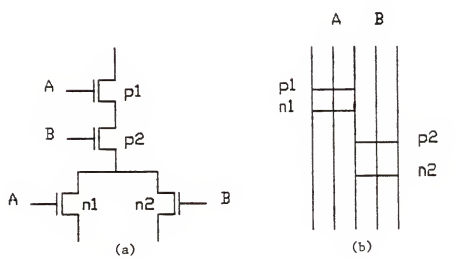


Figure 5.2 Device Locations on a Grid Structure. a) A Sample Circuit; b) Its Corresponding Grid Representation.

The second problem is then to determine an optimal permutation order for the device columns which results in the minimum total wiring length among the devices and signals.

5.1.1 Gate Splitting

Gate splitting is the process to convert the set of gate signals into a desired number of device columns. Let $G = \{ g_i \mid g_i$ is a gate signal $\}$ be a set of gate signals of the circuit, and each gate signal g_i be composed of a set of devices $g_i = \{ d_{i1}, d_{i2}, \ldots, d_{ik} \}$. Let N_g be the total number of gate signals. Let $G' = \{ g'_j \mid g'_j$ is a device column $\}$ be a set of device columns and each device column g'_j be composed of a set of devices $g'_j = \{ t'_{j1}, t'_{j2}, \ldots, t'_{jk} \}$. Let N_g and N'_g denote the total number of gate signals and device columns respectively. Then the Gate splitting process is to find the assignment function

$$f_7 : G \longrightarrow G'$$

such that

(1) The number of the device columns N'_g is equal to the number assigned by the circuit boundary analysis process.

(2) The number of total device rows n_r is minimized.

Given the equations described in Chapter IV,

$$n_r = \frac{N_d + WS}{n_c} \tag{4-29}$$

$$WS = \sum_{i=1}^{n_r} [N_{di} - MAX(N_{di})] \tag{4-30}$$

where n_c and n_r denote the number of device columns and device rows which are assigned during the cell boundary analysis process. Whenever there are more unused grid locations (WS), which are caused by uneven distribution of devices, there will be more device rows required. Consequently, the Gate splitting process will emphasize reducing the number of unused grid locations. From equation 4-30, it is obvious that WS can be minimized by minimizing the maximum number of N_{di}, i = 0 to N_g. The procedures of gate splitting are described as follows:

(1) Assign the initial device column number $N'_g = N_g$.

(2) Select a gate signal g_k where $N_{dk} = \text{MAX}(N_{di})$, i = 1 to N_g.

(3) Split the gate signal g_k into two device columns, g'_k and g''_k. The devices in gate signal g_k are partitioned into two subgroups. Since the numbers of n devices and p devices are equal within each device column, the number of devices in g'_k and g''_k will both be even numbers. Therefore, if N_{dk} of the original gate signal g_k has a divisor of 4, then

$$N'_{dk} = N''_{dk} = \frac{N_{dk}}{2} \tag{5-1}$$

or else

$$N'_{dk} = \frac{N_{dk} + 2}{2} \tag{5-2}$$

$$N''_{dk} = \frac{N_{dk} - 2}{2} \tag{5-3}$$

(4) Increase the number of N'_g by 1 and check if it is equal to the device column number n_c.

(5) If $N'_g = n_c$, stop. Otherwise, go to step 2.

5.1.2 Formulation of the Device Column Order Assignment Problem

The problem of device column order assignment can be considered as a classical Quadratic Assignment problem [47-49]. This problem is a generalization of the assignment problem in the following sense. Given n people (device columns) and a matrix [c_{ij}], where c_{ij} is a measure of the affinity (wires) between the two people i and j. Also are given n possible offices (slots) for these people and a matrix [d_{kl}], where d_{kl} is the distance between office k and l. If i is assigned to office p(i) and j is assigned to office p(j), then the cost of the assignment is taken to be $c_{ij} d_{p(i)p(j)}$. We now take as our measure of total cost the sum of $c_{ij} d_{p(i)p(j)}$ costs over all i and j, and seek the assignment which yields the smallest total cost. The problem can be formally described as follows [Problem - Quadratic Assignment] Given a cost matrix M = [c_{ij}], and a distance matrix D = [d_{kl}], find a permutation P of i,j such that

$$G = \sum_{ij} c_{ij} d_{p(i)p(j)} \tag{5-4}$$

is minimized.

Now we formulate the device column order assignment problem. Let T = { t_i | t_i is a transistor } be a set of transistors of the input circuit. Let G' = { g'_i | g'_i is a device column } be a set of device columns generated by the gate splitting process. Given a device t_i , let $S(t_i)$ denote a set of signals associated with the drain and source terminals of device t_i . For any device column $g'_i \epsilon G'$ let

$$S(g'_i) = \bigcup_{t_i \epsilon g'_i} S(t_i) \tag{5-5}$$

be the set of signals associated with the devices which are located on device column g'_i . The connections (cost function) between two device columns, g'_i and g'_j , are then defined by the common signals of the two device columns

$$S_{ij} = \left| S(g'_i) \bigcap S(g'_j) \right| \tag{5-6}$$

Let C = {1,2,....., n_c } be a set of column numbers. If the device columns g'_i and g'_j are to be assigned column numbers k and l, then the total horizontal wiring length of this single assignment is taken to be $S_{ij}(k-l)$. The distance matrix is therefore defined as D = [d_{kl}] where $d_{kl} = |k - l|$. Now the device column order assignment problem can be formally described as

[Problem - Device Column Order Assignment] Given a connectivity func-

tion S = [S_{ij}] and a distance function D = [d_{kl}], find a permutation order among the device columns G = { $g'_1, g'_2, \ldots$ } such that the total horizontal wiring length

$$HW = \sum_{ij} S_{ij}\, d_{p(i)p(j)} \tag{5-7}$$

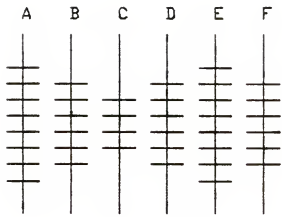
is minimized, where p(i) denotes the column order assignment of ith device column g'_i. Figure 5.3 depicts the problem of device column order assignment. The device columns are represented as interconnected modules. The modules are then to be linearly placed in an equivalent number of slots (columns) with an emphasis to minimize the total cost function.

5.1.3 Clustering Algorithm

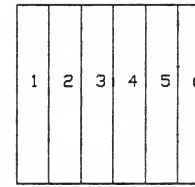
The quadratic assignment problem has been proved to be NP-complete [47]. Several researchers [47-49] have proposed heuristic algorithms in an effort to generate near-optimal results. In 1972, Shuler and Ulrich proposed a clustering algorithm [50] to solve the quadratic assignment problem. In this dissertation, an algorithm similar to Shuler and Ulrich's approach has been adopted to solve the device column order assignment problem for the following reasons:

(1) For device placement problems, the cost function contains more than just the connection and distance function. Transistor abutment should also be considered. The clustering algorithm has the capability to handle combined cost function.

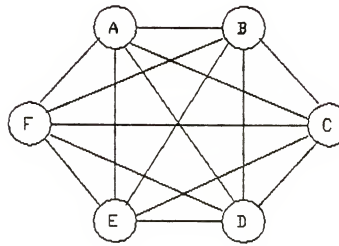
(2) The clustering algorithm is free to choose any signal as the seed signal so as to match the cell boundary signal requirement.



(a)



(c)



(b)

Figure 5.3 The Device Column Order Assignment Problem. a) Grid Representation of a Circuit; b) All the Device Columns Are Represented as a Set of Interconnected Modules. Each Module (Circle) Represents a Device Column; c) Available Sites for the Placement of the Device Columns.

(3) The algorithm is easy to implement and the computation time is short.

The Clustering technique was first implemented by Shuler and Ulrich for the linear placement of a set of interconnected modules. The idea of clustering is to analyze a set of interconnected modules and form groups that have strong internal connections but weak external connections to other groups. Because the strongly connected modules are placed together while the far apart modules have weak interconnections, the total cost is expected to be optimized. Figure 5.4 shows several different permutations for the placement of three modules. Assume that there are 5 connections between modules A and B, 3 between B and C, and 2 between A and C. In case 1, the strongly connected modules A and B, B and C are placed together while A and C are placed apart. If we assume that the distance between each pair of adjacent modules is 1. Then the total cost (wiring length) for case 1 is 12. In case 2, modules A and B are still placed together but modules B and C are now placed apart. Since S_{BC} is greater then S_{AC} , the longer distance between B and C increases the cost to 13. In case 3, the situation is even worse because the strongly connected modules A and B are now placed apart. The total cost in this case is 15.

The clustering value. Clustering is done in a pairwise fashion. A clustering value is calculated for each pair of interconnected elements where an element can be either a separate module or a cluster of modules. The clustering value serves as the strength measure for the interconnection relationship among all the elements. The value should be a function of both the conjunctivity (connections between i and j) and disjunctivity (connections not between i and j); therefore, even though the pairs of two elements have the same number of connections, their clustering value may be quite different. Figure 5.5 shows the factor of disjunctivity for several different nets. In Figure 5.5(a) and 5.5(b), both pairs

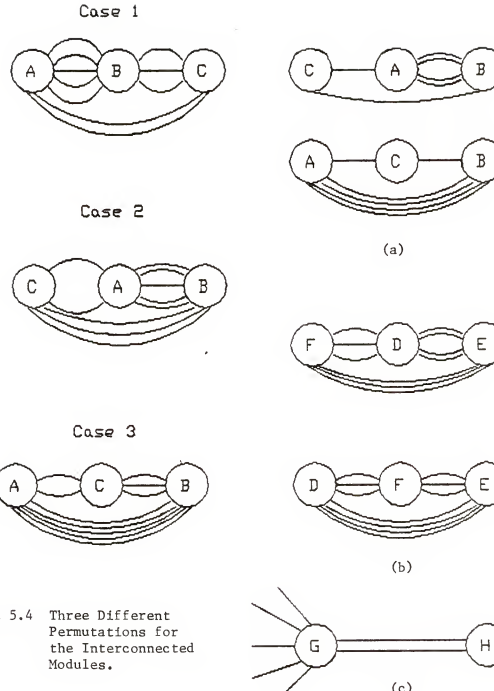


Figure 5.4 Three Different Permutations for the Interconnected Modules.

Figure 5.5 The Importance of Disjunctivity in Clustering. a) The Four Connections between A and B Are Significant; b) The Four Connections between D and E Are Not So Significant; c) Module H Must Be Connected with G.

have the same number of connections. However, if the elements A and B in Figure 5.5(a) are not clustered, the total cost will increase from 7 to 10. In Figure 5.5(b), if elements D and E are not clustered, the cost will increase from 13 to 14. Therefore, the clustering of A and B is more important than the clustering of D and E. In Figure 5.5(c), element G does not tend to be clustered with element H because G has five connections to other elements, but H has the only choice to be clustered with element G. Consequently, the clustering value SV_{ij} for elements i and j is given by the following equation

$$SV_{ij} = \frac{S_{ij}}{T_i - S_{ij}} + \frac{S_{ij}}{T_j - S_{ij}} \tag{5-8}$$

where

S_{ij} = connections between elements i and j

T_i = total connections to element i

T_j = total connections to element j

The clustering procedures. After the clustering value is calculated for all connected pairs of elements, the pairs with the greatest CV's are clustered. The clustered pairs become new elements which are then eligible for the next pair-wise clustering. The process is continued until all modules (device columns) are drawn into a single cluster or the remaining clusters are independent.

The clustering tree. After the clustering procedure, a binary tree, which is called the clustering tree, is generated. The clustering tree represents the topological relationships among the modules and clusters. An example is given in Figure

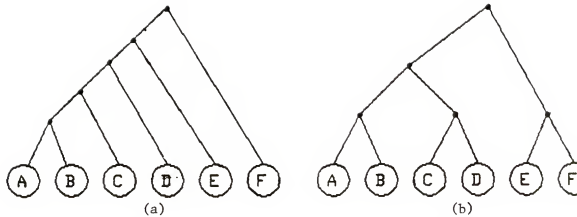


Figure 5.6 Two Kinds of Clustering Trees. a) All Modules Are Added to the Same Cluster One by One; b) Small Clusters Are Created First.

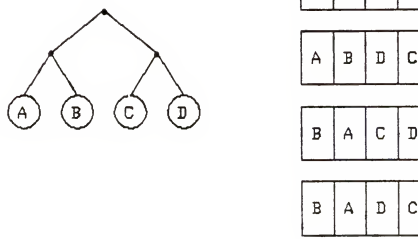


Figure 5.7 Four Possible Linear Placement Orders for the Four-Module Cluster.

5.6. The binary tree is a collection of elements called nodes along with a relation (parenthood) that places a hierarchical structure on the nodes. A single node by itself is a subtree. The parenthood relationship is represented by the lines downward from a node n to other nodes $n_1, n_2, \ldots, n_k$. Node n is called the parent of nodes $n_1, n_2, \ldots, n_k$. Nodes $n_1, n_2, \ldots, n_k$ are called the children of node n. The nodes which have no children are called leaf nodes. The node which locates at the top level of the binary tree (or subtree) is called the root. The level of a node in the tree is the length of a longest path from the root to the node. As shown in Figure 5.6, each module (device column) is represented as a leaf node at the bottom level. Whenever two nodes are clustered, the new cluster is represented as a parent node of those two nodes. The clustering process continues until all the modules are clustered as a single cluster which is represented as the root node. The topological information included in the binary tree is that the two children of the same parent should be placed adjacently. Figure 5.7 shows the possible linear placement of a cluster.

The factor of transistor abutment. The clustering value function discussed so far only concerns about the connections among the elements. However, transistor abutment is also an important factor which should be considered in device placement. Figure 5-8 demonstrates the importance of transistor abutment in device placement. In Figure 5.8(a), an extra diffusion grid is needed because transistor A and B cannot be abutted by this arrangement. In Figure 5.8(b), the extra diffusion grid is removed because A and B can share the same diffusion area.

Now we try to define a clustering value which also considers the factor of transistor abutment. Let M_i and M_j denote two clusters which are represented by two subtrees T'_1 and T'_2 respectively. Let m_{ip} denote the pth module contained in cluster M_i , and m_{jq} denote the qth module contained in cluster M_j . Let s_{ipjq}

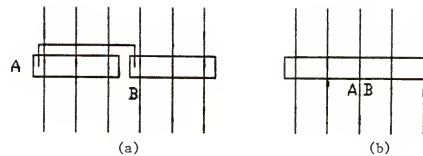


Figure 5.8 Importance of Transistor Abutment. a) Two Devices Are Not Abutted; b) Abutment Reduces the Number of Grid Columns and Connections.

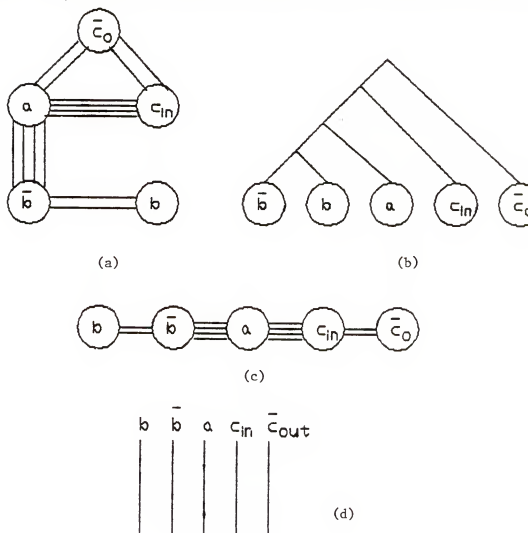


Figure 5.9 Assignment of the Device Column Orders of the Circuit in Figure 3.5. a) Graph Representation of the Device Columns; b) The Generated Clustering Tree; c) Result of Linear Placement; d) The Final Device Column Orders.

denote the number of connections between module m_{ip} and m_{jq} . Let l_{ip} denote the level of module m_{ip} within the subtree T_1 . Now the expected number of transistor abutment between cluster M_i and M_j can be represented as

$$ABT_{ij} = \sum_{p=1}^{m} \sum_{q=1}^{n} \frac{s_{ipjq}}{2^{l_{ip}} 2^{l_{jq}}} \tag{5-9}$$

By adding the factor of equation 5-9 into equation 5-8 yields

$$SV_{ij} = \frac{S_{ij}}{T_i - S_{ij}} + \frac{S_{ij}}{T_j - S_{ij}} + \sum_{p=1}^{m} \sum_{q=1}^{m} \frac{s_{ipjq}}{2^{l_{ip}} 2^{l_{jq}}} \tag{5-10}$$

Linear placement algorithm. The linear placement algorithm uses the binary tree generated by the clustering algorithm. For a one-dimensional row, any two adjacent clusters can shift positions. Therefore, the number of possible linear arrangement is

$$NP = 2^{n-1} \tag{5-11}$$

If two symmetrical placements are considered as equal, the number is only half

$$NP = 2^{n-2} \tag{5-12}$$

This number is much smaller than the total possible linear placement arrangement without binary tree structure which is

$$NP = n!/2 \tag{5-13}$$

Shuler [50] proposed two possible binary-tree shuffling methods to perform the linear placement of the clustering result. Although he proves that the top-down linear placement, which shuffles from the highest level of nodes of the binary tree, achieves a better result (with shorter total wiring length), it is not appropriate for gate column ordering. This method neglects all the transistor abutment factors and causes poor result. The method we adopt here is a revised bottom-up method which considers the factor of transistor abutments during binary tree shuffling. The solution is to proceed from the most basic subtrees (a pair of nodes) and always link two already linearly placed subtrees into a single linear placement. Consider the two subtrees which are arranged linearly with m and n nodes respectively in each subtree:

(1,2,....,m) (1,2,....,n)

Since the two linear placements cannot be intermixed by the hypothesis, there are only four ways the two linear placements can be linked:

(1,2,....,m) (1,2,....,n)
(1,2,....,m) (n,....,2,1)
(m,....,2,1) (1,2,....,n)
(m,....,2,1) (n,....,2,1)

The linear placement process then starts from a leaf node. Each time the clustering value of all four different linear placements between two adjacent clusters are calculated. The placement with the maximum clustering value is selected and the algorithm proceeds until all the modules have been added to the linear placement. Figure 5.9 shows the device placement result for the "compare circuit" shown in Figure 3.5.

5.2 Device Orientation Assignment

In section 5.1, we present a technique to determine the horizontal grid locations of the devices. However, the horizontal grid locations of the device terminals are still not fixed because the drain and source terminals of each device are free to switch positions. Devices which have different source/drain positions are termed to have 'different orientations'. Let us call that a device is east oriented if the drain terminal is to the east of the source terminal and vice versa. Because a net is interconnected through device terminals, different arrangements of device orientations can cause much different wiring results. The task of Device Orientation Assignment is to find an optimal set of device orientations such that the total wiring length of the cell layout is minimized.

The computational complexity for exhaustively searching an optimal set of device orientations is $O(2^N)$ where N is the total number of devices. However, such an exhaustive search is meaningless because the vertical wiring length is still unknown at this stage. In this dissertation, we use a net processing technique to decide the orientations of some of the devices. The orientations of the other devices are then determined by heuristic algorithms which emphasize reducing the horizontal wiring length and vertical wiring length.

Net processing. Each net in the circuit is defined as an electrical common

point for several transistor terminals. Since all the transistors in this net are to be interconnected by wire segments, it is advantageous to place all those transistors at the same device row. In that case, the interconnections usually can be implemented as a single horizontal wire segment. However, as each device contains two terminals which may belong to two different nets, the transformation from the electrical nets into net segments is not direct.

Net splitting. In case a net in the circuit can not be assigned as a single horizontal line segment, it should be split. This situation happens if two transistors in the net are located at the same column or three transistors in the net are located at adjacent columns. Figure 5.10(a) shows an example of net splitting.

Net merging. This procedure will increase the number of transistor abutments. Two nets will be merged as a single horizontal line segment if these two nets have a transistor in common and they are located at different sides as shown in Figure 5.10(b).

5.2.2 Heuristic Algorithms

The device orientation assignment process utilizes several circuit heuristics to determine the device orientations. The circuit heuristics, sorted in priority order, are described as follows:

(1) Maximizing the number of transistor abutments.

(2) Maximizing the number of vertical wiring alignments.

(3) Minimizing the number of vertical obstructions.

(4) Minimizing the total horizontal wiring length.

All these circuit heuristics are described in the following paragraphs.

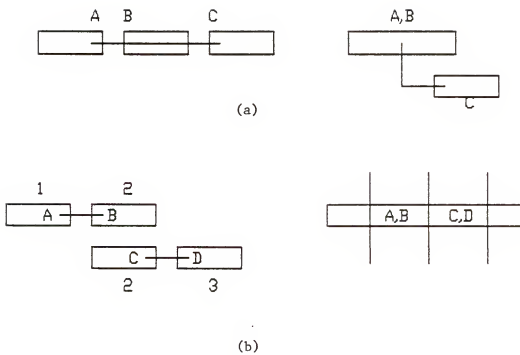


Figure 5.10 Two Tactics of Net Processing. a) Net Splitting; b) Net Merging.

Maximizing the number of transistor abutments. Transistor abutment in cell layout not only reduces the device wiring length but also reduces the device area. Consequently, it is the most important factor to be considered. There are two essential conditions for two transistors A and B to be abutted:

(1) A and B must be placed at adjacent columns (or rows).

(2) A and B must have at least one connection.

(3) A and B must be properly oriented such that the interconnected terminals of A and B are at adjacent locations.

Figure 5.11(a) shows that transistors T1, T2, and T3 can be abutted by proper arrangement of those three transistors.

Maximizing the number of vertical wiring alignments. If two transistors are located at adjacent device rows, it is also advantageous to arrange the device orientations such that the interconnected terminals locate adjacently. In such case, the two transistors can be either vertically abutted or connected by a vertical diffusion run. This is illustrated in Figure 5.11(b).

Minimizing the number of vertical obstructions. If a vertical connection between any two device terminals is blocked by an obstacle, it requires extra wiring detour to bypass the obstacle. As shown in Figure 5.11(c), the transistor should be properly oriented, if possible, to avoid vertical obstruction.

Minimizing the total horizontal routing length. Different orientation assignments among two horizontally connected devices usually cause different wiring lengths. As shown in Figure 5.11(d), we can see that if a device is right connected, it is better to place the interconnected terminal at right. Otherwise, it should be placed at left. If a device has both right and left connections, we consider it as right (left) connected if the number of right (left) connections is greater than the left (right) connections.

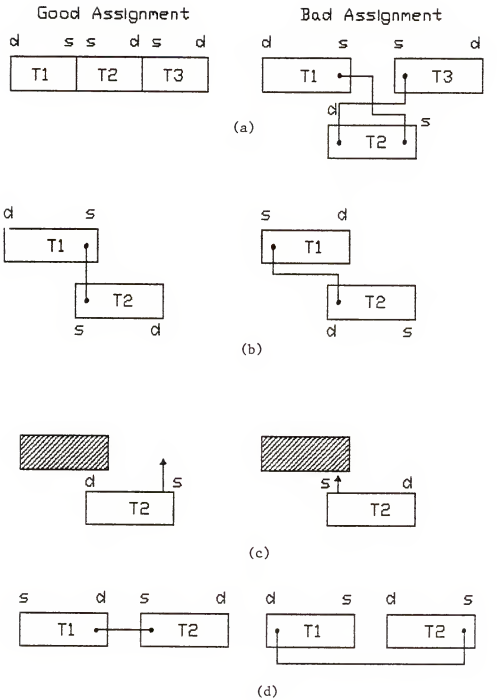


Figure 5.11 Heuristic of Device Orientation Assignment. a) By Transistor Abutment; b) By Vertical Alignment; c) By Vertical Obstruction; d) By Horizontal Wiring Length.

5.3 Device Row Roder Assignment

After all the device orientations are determined, the last step for device placement is to determine the vertical grid locations (row orders) of the devices. The permutation number for the exhaustive search of optimal device row orders is also very huge. We assume that there are n columns for a cell layout. Each column has m device rows. Let N_{di} denote the number of devices in ith device column. Since the devices can be permuted only within each device column, the total permutation for the device row orders is

$$PN = \sum_{i=1}^{n} C(m, N_{di}) N_{di}! \tag{5-14}$$

Fortunately, the computational complexity has been reduced tremendously by the net processing technique described in the previous section. After the net processing, the devices are converted into a set of horizontal wire segments with possible vertical connections. The device placement problem is, therefore, converted into the wire segment placement problem. Since the number of horizontal wire segments derived from the net processing procedures is far less than the total number of devices, the computational complexity is considerably reduced.

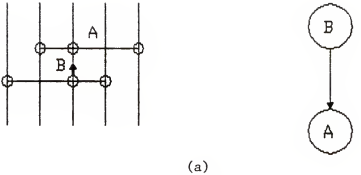
Placement of a set of horizontal wire segments is a classical channel routing problem. The algorithm proposed by Hashimoto and Stevens [51] was proved to be optimal. However, this algorithm did not consider the vertical connections associated with the wire segments. In this dissertation, we used a constraint graph-based approach to handle the wire segment placement problem with vertical connections.

The (vertical) constraint graph is defined as a directed graph which implicitly determines the placement order of the horizontal wire segments. The constraint graph CG consists of a set of vertices V = { v_1, v_2 ,..}, a set of edges E = { e_1, e_2 ,...}, and a mapping Ψ that maps every edge onto some ordered pair of vertices (v_i, v_j). The vertex v_i , which the edge e_k is incident out of, is called the initial vertex. The vertex v_j , which the edge e_k is incident into, is called the terminal vertex of e_k . A directed edge (v_i, v_j) is represented by a line segment between v_i and v_j with an arrow directed from v_i to v_j . Each horizontal wire segment to be placed is represented as a vertex in the digraph. A directed edge (v_i , v_j) indicates that horizontal wire segment v_i should be placed above v_j .

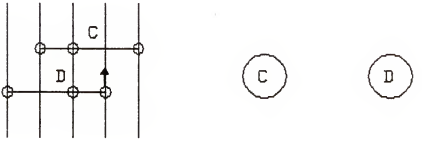
Constraint graph and binary relations. Due to the wiring considerations, a binary relation R between vertex pairs (v_i , v_j) always exists. We write

$$v_i \; R \; v_j$$

and say that v_i has relation to v_j . There are three kinds of binary relations between the vertex pairs. Figure 5.12 illustrates all the binary relations. In Figure 5.12(a), both of nets A and B have a vertical, upward connection. However, the vertical connection of B has the same location of a device terminal in A. In this case, B is vertically obstructed by A. It is obvious that whenever B is vertically obstructed by A, B should be placed above A. Otherwise, the vertical connection of B will need extra wiring detour to bypass the blocking obstacle. Consequently, we define *ABOVE* and *BELOW* relations between a pair of vertices (v_i, v_j) with v_i *ABOVE* v_j indicates that v_i should be placed above v_j and vice versa. Another relation v_i *UNREL* v_j indicates that v_j and v_j are not blocking each other and have placement constraint between them. Figure 5.12(b) shows that nets C and D can be placed either way because they have no



(a)



(b)

Figure 5.12 Binary Relations Between Two Nets. a) Net B is ABOVE Net B; b) Net C and Net D are not Related.

wiring constraints.

Construction of the vertical constraint graph. Since each net is composed of a set of horizontally interconnected devices, the binary relation R between any two nets A and B is determined by the binary relations among the devices in A and B. Let R' denote the binary relation between two devices t_i and t_j . The definition of binary relations *ABOVE* , *BELOW* and *UNREL* between two nets is also applied to the device relations. Let t_i *ABOVE* t_j indicate that device t_i must be placed above t_j because of vertical obstruction. Suppose that net n_i contains a set of devices $n_i = \{ t_{i1}, t_{i2}, \ldots, t_{ik} \}$ and that net n_j contains a set of devices $n_j = \{ t_{j1}, t_{j2}, \ldots, t_{jl} \}$. Let us define a special binary relation *ALIGN* between two devices. In which case, t_i *ALIGN* t_j indicates that device t_i must be placed at the same row as t_j . Since all the devices in the same net will be placed at the same row, the following binary relations hold for net n_i and net n_j :

$$
\begin{array}{c}
t_{i1}\ ALIGN\ t_{i2} \\
t_{i2}\ ALIGN\ t_{i3} \\
\cdot \\
\cdot \\
t_{ik-1}\ ALIGN\ t_{ik} \\
t_{j1}\ ALIGN\ t_{j2} \\
t_{j2}\ ALIGN\ t_{j3} \\
\cdot \\
\cdot \\
t_{jl-1}\ ALIGN\ t_{jl}
\end{array}
$$

The *ALIGN* relation has the property that if for any three transistors t_i, t_j,

t_k,

$$t_i \ ALIGN \ t_j \text{ and } t_i \ R \ t_k$$

always implies

$$t_j \ R \ t_k$$

where R can be either one of the four binary relations described above. All four binary relations also have the transitivity property such that,

$$t_i \ R \ t_j \text{ and } t_j \ R \ t_k$$

always imply

$$t_i \ R \ t_k$$

Now let us describe how to transform devices' binary relations into nets' binary relations. Let $t_{ik} \epsilon n_j$ and $t_{jl} \epsilon n_j$. Then the binary relations *ABOVE* and *BELOW* have the following properties:

$$\text{IF } t_{ik} \ ABOVE \ t_{jl} \text{ THEN } n_i \ ABOVE \ n_j$$

$$\text{IF } t_{ik} \ BELOW \ t_{jl} \text{ THEN } n_i \ BELOW \ n_j$$

By using these two properties and the device relation properties described previously, the constraint graph of the nets can be transformed from the

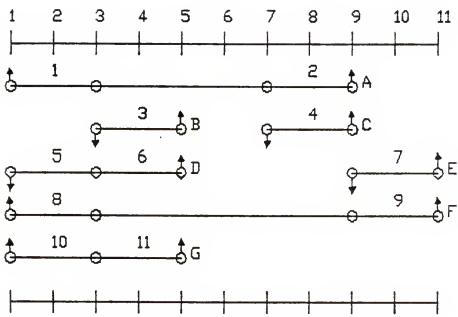


Figure 5.13 Netlist Representation of a Circuit.

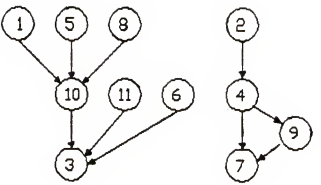


Figure 5.14 Vertical Constraint Graph of the Devices.

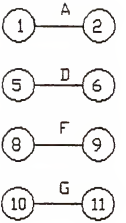


Figure 5.15 The ALIGN Relation.

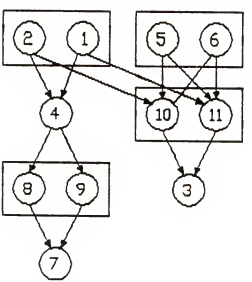


Figure 5.16 Vertical Constraint Graph of the Devices after the Binary Operations.

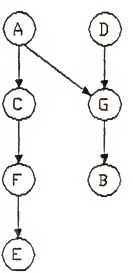


Figure 5.17 Vertical Constraint Graph of the Nets.

constraint graph of the devices. Figure 5.13 shows an example of net-list representation which is generated by the net processing process. The vertical constraint graph of the devices, which is shown in Figure 5.14, is constructed by the vertical connections and relative positions among the device terminals. Figure 5.15 depicts the *ALIGN* relations among the net segments. Figure 5.16 shows the constraint graph of the devices after applying transitivity properties and *ALIGN* relation to the devices. An equivalent constraint graph for the nets is then constructed accordingly as shown in Figure 5.17.

Constrained left-edge-first algorithm. After the vertical constraint graph for the nets is constructed, an algorithm called the constrained left-edge- first is then applied to place the horizontal wire segments. The algorithm is a modification of Hashimoto and Steven [51] and Kuh et al. [52] approaches. This algorithm uses a merging technique to merge two or more nets into a single net while observing the layout constraint. The wire segments in the final constraint graph after the merging process can be easily placed one by one into the device rows. The procedures of the device row order assignment can be summarized as:

(1) Construct the constraint graph of the devices.

(2) Construct the constraint graph of the nets based upon result of (1).

(3) Sort the edges in net segment set W based upon the value of the left coordinate of each net segment. If several net segments have the same left edge, the net segment with the higher priority in the constraint graph will be placed on top.

(4) Select the first net segment W_1 in W and delete this net segment from W.

(5) Find all the net segments which have left edge coordinates smaller than

that of the net segment W_1 and store all those net segments into a list L.

(6) Select a net segment W_i from L which (a) is closest to the right of W_1 and (b) minimizes the increase of the longest path of the vertical constraint graph.

(7) Merge net segments W_1 and W_i . Delete the net segment W_i from W .

(8) Repeat steps of (6) and (7) until no nets in L can be merged with net W_1 .

(9) Go to step 4 and repeat steps 5 to 8 until W is empty.

(10) Place the net segments one by one according to the priority order in the final merged constraint graph.

The device row order assignment procedures are illustrated by an example shown in Figure 5.18. Figure 5. 19 shows the intermediate steps of the graph merging process. The final implementation of the net segments, according to the merged constraint graph, is shown in Figure 5.20

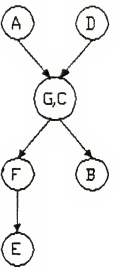


Figure 5.18 Intermediate Result of Net Merging.

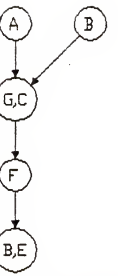


Figure 5.19 Final Result of Net Merging.

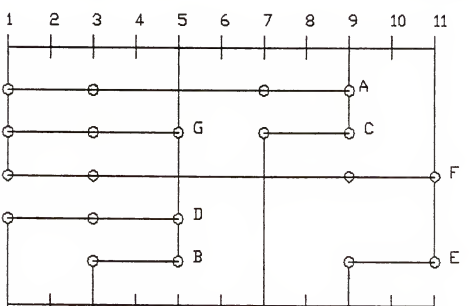


Figure 5.20 Placement of the Net Segments According to the Vertical Constraint Graph in Figure 5.19.

CHAPTER VI
ROUTING IN HIGH DENSITY GATE MATRIX STRUCTURE

After the device placement procedures, the circuit representation can be transformed from circuit domain into topological domain with device terminals represented as grid points and gate and power signals represented as grid lines. The signal nets, which are originally defined as sets of electrically common device terminals, are now defined as sets of grid points and lines which should be connected together. In this chapter, we present algorithms which realize the circuit connectivity among the grid points and lines with an emphasis to minimize the total wiring length under all the routing constraints.

Six sections are included in this chapter. Section I describes the general routing constraints under the Gate Matrix Layout and CMOS IC design environment. Section II depicts the terminal labeling techniques which generate important data structures for the use of wire segment generation and detailed routing processes. A method for the generation of wire segments is given in Section III. The method converts a muti-point net into a set of two-point wire segments (wire-list) which has the minimum total wiring length. The minimum spanning tree algorithm is used to find the shortest path which interconnects the multi-point net. Section IV presents the routing techniques to find the paths for the generated wire segments. The system combines several channel routing techniques to execute detailed routing for different routing schemes. Finally, the last section describes the techniques for the improvement of cell layout and for small-scale modification of cell configuration.

6.1 The Routing Constraints and Assumptions

The routing constraints can be considered as a prescription for preparing the photomasks that are used in the fabrication of integrated circuits. The constraints provide a necessary communication link between the circuit design and fabrication process. All the constraints are derived from the observation of the interactions between different mask layers.

Processing technology and layer representation. The advances in CMOS processing technology has resulted in the complex CMOS processes which somewhat inhibit the visualization of the mask levels in the actual fabrication process. Nevertheless the design process can be abstracted to a manageable number of conceptual layout levels that represent the key features at the symbolic level. In this dissertation, a single-metal, twin-well, poly-gate CMOS technology is assumed. Only five most important mask layers, which includes n diffusion, p diffusion, polysilicon, metal, and contact, of the CMOS fabrication process are considered at the symbolic level. Each mask layer is represented as a stipple pattern in schematic layout or a special symbol in computing process. Figure 6.1 shows several frequently used stipple patterns.

Due to the interactions between different mask layers, there are several constraints for the interconnection path within the cell. Those constraints are called the routing constraints. Following is the list of the routing constraints for symbolic cell layout.

(1) The metal layer and polysilicon layer are isolated according to natural material property. Contacts are, therefore, needed for the the interconnection between those two layers. The same situation applies for the layers of metal and diffusion.

(2) Crossover is allowed between (a) metal and diffusion layers (b) metal

Figure 6.1 Major Stipple Patterns Used in Schematic Layouts.

and polysilicon layers.

(3) Crossover is not allowed between (a) two metal layers (b) metal and contact (c) diffusion and contact (d) two diffusion layers (e) diffusion and polysilicon layers.

The routing constraints described above are illustrated by Figure 6.2.

Basic assumptions for Gate Matrix routing. Based upon the described routing constraints, several assumptions have been imposed on the routing process with an emphasis to reduce the potential routing conflicts and to simplify the algorithm design.

(1) Polysilicon is used to realize gate signals. It runs vertically and is separated by diffusion runs.

(2) Wire connections can be realized as either metal or diffusion layer. Metal is used for horizontal wire connection. Diffusion is used for horizontal wire connection. The only exception is that wire connection at the left-most side and right-most column can be realized as metal lines.

(3) The power net, which is realized as a horizontal line, is placed at the top of the cell. The ground net is placed at the bottom of the cell. Although both nets are represented as a set of separated grid points, all those points are considered as interconnected implicitly. Device terminals which are connected to the power (ground) net can be connected to any grid point of the net.

(4) A polysilicon gate column is also considered as a set of implicitly interconnected grid points. Each device terminal which is connected to the gate column can be connected to any one of the grid points.

(5) Horizontal (metal) wires can be placed on either device rows or horizontal routing channels.

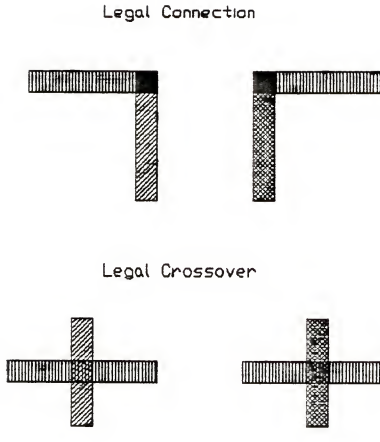


Figure 6.2 Two Basic Layout Rules.

(6) Vertical (diffusion) wires which need to run over any device row require a through terminal on that device row.

(7) We assume that unlimited horizontal routing tracks can be inserted between two adjacent device rows or between a device row and the power (ground) net. The insertion of the vertical routing tracks, however, is strictly limited. It is permitted only when the system, after boundary analysis, suggests to do so or channels are congested.

6.2 Terminal Labeling

The task of terminal labeling is to convert the information of circuit connectivity from the circuit domain to topological domain. Such information is needed for the use of wire segment generation and detailed routing. Let $S = \{S_i \mid S_i \text{ is a net}\}$ be a set of nets which describes the circuit connectivity. Each net S_i consists of a set of device terminals $S_i = \{p_{i1}, p_{i2}, \ldots, p_{ik}\}$. After the device placement procedures, the grid location of each device terminal is determined. Consequently, we can describe the net connectivity as a different format. Let $S' = \{S'_i \mid S'_i \text{ is a net}\}$ be a set of nets which describes the circuit connectivity. Then we can describe each net S'_i as a set of grid points $S'_i = \{p'_{i1}, p'_{i2}, \ldots, p'_{ik}\}$ where $p'_{ik} = (x_{ik}, y_{ik})$, is the grid location assigned for device terminal p_{ik}. We now describe the terminal labeling process.

Let $S = \{S_1, S_2, \ldots, S_s\}$ be the set of nets. Each net S_k is given a net number k over integers 0,1,2,...,s where s is number of nets. We then assign the same net number k to all the device terminals p_{ik} which is contained in the net S_k. Let N be the number of grid rows and M be the number of grid columns on the Gate Matrix structure. Consequently, there are MN grid location available for the use of device terminals. We call each grid location as a "terminal site". There are

two kinds of terminal sites. If a terminal site is occupied by a device terminal, it is called a "used terminal site" and is assigned the net number of that device terminal. Otherwise, the terminal site is called "unused terminal site" and is assigned a number 0. We call the terminal table as the topological net representation. Figure 6.3 shows the topological net representation of a sample circuit. Each topological net representation includes two kinds of key information

(1) All the terminal sites with the same net number represent a multi-point net which should be interconnected by wire segments.

(2) The distance between two net terminals can be measured as the distance between the two grid points on which the net terminals locate.

(3) The terminal sites labelled as "0" represent unused terminal sites. Only the unused terminal sites can be assigned as through terminals which allow the vertical diffusion path to pass through.

After the terminal labeling process, the circuit connectivity is successfully transformed into the topological domain. Each net $S'_i = \{p'_{i1}, p'_{i2}, \ldots\ldots, p'_{ik}\}$ is represented as a set of grid points instead of a set of device terminals. The task of the routing process is now to interconnect all the grid points within a multi-point net. Figure 6.4 shows the final routing result of Figure 6.3.

Obstruction map. Besides the topological net representation, we also need an obstacle map to indicate which mask layers are located on each terminal site. This information is essential because the device rows, which contain a lot of routing obstacles, are also used as part of the routing channels. The obstacle map will indicate which parts of the device rows can or cannot be used as routing channels. Initially, the obstacle map contains only the information of the device terminals. As routing process proceeds, the information of the wire

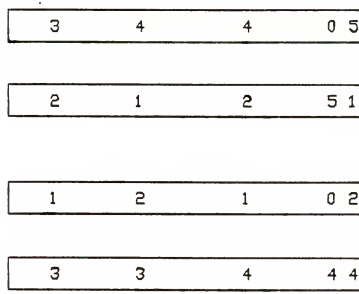


Figure 6.3 Terminal Labelling of a Sample Circuit.

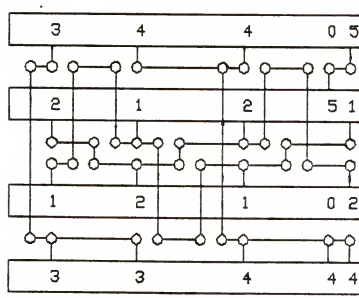


Figure 6.4 Final Routing Result of Figure 6.3.

segments and contacts will be gradually added to the map. The obstacle map indicates what kind of layers are located on each terminal site and implicitly indicates which kinds of wire segments or contacts are not allowed to passed through or placed on that terminal site. To implement the routing algorithms, we provide a M-flag to represent the information of mask layers on each terminal site:

M-flag:

M(x,y) = 0 indicates an unused terminal site at (x,y)
M(x,y) = 1 indicates polysilicon layer at (x,y)
M(x,y) = 2 indicates a device terminal at (x,y)
M(x,y) = 3 indicates both polysilicon and diffusion at (x,y)
M(x,y) = 4 indicates metal layer at (x,y)
M(x,y) = 5 indicates both metal and polysilicon at (x,y)
M(x,y) = 6 indicates both metal and diffusion at (x,y)
M(x,y) = 7 indicates metal, polysilicon, and diffusion at (x,y)
M(x,y) = 8 indicates a contact at (x,y)
M(x,y) = 9 indicates diffusion run at (x,y)

6.3 Wire Segment Generation

Most routing algorithms deal with the problem of connecting two entities, such as two points or a point and a wire. Hence, given an n point net, $S'_i = \{p'_{i1}, p'_{i2}, \ldots, p'_{in}\}$, the net is usually divided into n-1 wire segments, each of which connects two points (p'_{ij}, p'_{ik}). Three major wire segment generation approaches [53-64] have been proposed since the late 1950s. The first is to chain the n points, i.e., to find a permutation p of the signal points 1.2,....n so that

$$\sum_{i=1}^{n-1} d_{p(i),p(i+1)}$$

is minimum, where p(i) is the ith point in the permutation and $d_{p(i),p(i+1)}$ is the distance between point p(i) and p(i+1). Unfortunately, this problem is equivalent to the traveling salesman problem [53-58] and is NP complete. The second most common technique used for wire segment generation is to construct a minimal spanning tree [59-62], which can be done in time no worse than $O(n^2)$ [62]. The third technique is to form a Steiner tree [63-64], which is also proved to NP complete. All three tree structures for wire connection are shown in Figure 6.5.

Among the three tree structures, the Steiner tree usually results in the shortest total wiring length while minimum spanning tree is ranked the second and chain connection the third. Gilbert and Pollak [64] has conjectured the relationships among the optimal results of the three tree structures which can be specified as the following inequality:

$$L_S < L_M < L_T < 2L_S$$

where L_S denotes the length of the minimal Steiner tree, L_M denotes the length of the minimal spanning tree, and L_T denotes the minimal length of the traveling salesman tour. The average ratio L_S/L_M, according to Gilbert and Polak's [64] estimation, is about 0.866.

As the Steiner tree can result in the shortest total wiring length, it is supposed to be an ideal method for wire-list generation. However, since there are no known existing algorithms which is able to find the Steiner tree with more than five vertices, we choose minimum spanning tree as the tree structure

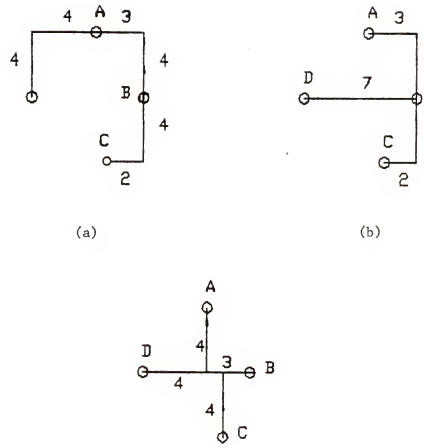


Figure 6.5 Three Different Structures for Wire Connections. a) Chain Structure; b) Minimum Spanning Tree; c) Steiner Tree.

for wire segment generation. Special routing techniques are then used in final routing process to improve the result.

6.3.1 The Minimum Spanning Tree Algorithm

A tree T is said to be a spanning tree of a connected graph if T is a subgraph of G and T contains all vertices of G. Since the wire segment generation problem is to find a shortest path which interconnects all the grid points, it can be conceptually modeled as a minimum (shortest) spanning tree problem. Given a n point net $S'_i = \{p'_{i1}, p'_{i2}, \ldots, p'_{in}\}$ and a distance matrix $D = [d_{jk}]$, where d_{jk} is the distance between point p'_{ij} and p'_{ik}, we represent the n point net as a weighted graph G which has n vertices and C(n,2) edges. Each vertex v_j represents the grid point p'_{ij} and each edge $e_{jk} = (v_j, v_k)$ represents a path between vertex v_j and v_k. The weight of an edge e_{jk} is equal to the distance d_{jk}.

There are several methods [59-62] available for actually finding the minimum spanning tree. One algorithm due to Kruskal [59] has been chosen and is described as follows:

(1) Sort all edges of the graph G in order of nondecreasing weight and store them in a list L.

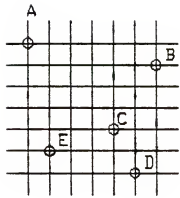
(2) Select the smallest edge from L. Store it into another list L' and remove it from L.

(3) Select another smallest edge from L that makes no circuit with the previously selected edge. Store it into L' and remove it from L.

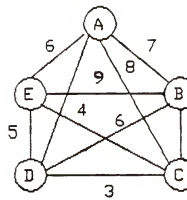
(4) Continue step 3 until n-1 edges have been selected.

(5) All the edges in L' constitute the minimum spanning tree.

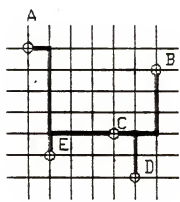
Figure 6.6 uses an example to illustrate the minimum spanning tree generation process. Figure 6.6(b) shows the weighted graph converted from the



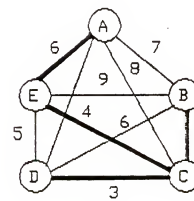
(a)



(b)



(c)



(d)

Figure 6.6 Minimum Spanning Tree Generation Process. a) A Multi-Point Net; b) Its Corresponding Weighted Graph; c) Minimum Spanning Tree on Grids; d) Minimum Spanning Tree on the Graph.

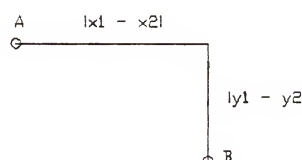


Figure 6.7 The Rectilinear (Manhattan) Distance between Point A and Point B.

multi-point net depicted in Figure 6.6(a). The minimum spanning tree of the weighted graph is generated by computer and is shown by bold lines on Figure 6.6(d). The corresponding minimum spanning tree on the grid structure is shown as interconnected wire segments in Figure 6.6(c).

6.3.2 Distance Estimation in Gate Matrix Structure

Since the generation of minimum spanning tree is based upon the comparison of the distance functions among the grid points, the accuracy of the distance measure is very critical to the success of a minimum spanning tree algorithm. For the past two decades, rectilinear distance (or called Mahattan distance) has been the most popular distance measure function used in layout algorithms. As shown in Figure 6.7, the The Manhattan distance between two points (x_1,y_1) (x_2,y_2) is simply

$$DST(A,B) = \left|x_1 - x_2\right| + \left|y_1 - y_2\right| \tag{6-1}$$

However, in the Gate Matrix layout environment, the distance function between two objects (can be either points or lines) varies in different situations.

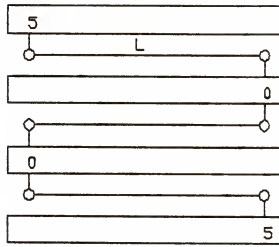
(1) Distance between two gate signals. Because each split gate signal is represented as a vertical polysilicon grid line in the Gate Matrix, net connection between two gate signals will be topologically represented as a horizontal wire segment between two vertical lines. Let the horizontal coordinate of the line A be x_1 and horizontal coordinate of line B be x_2. Then the distance measure between these two lines can be represented as

$$DST(A,B) = \left| x_1 - x_2 \right| \tag{6-2}$$

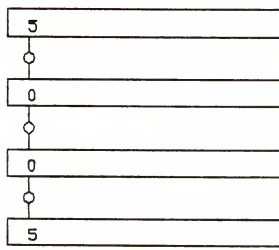
(2) Distance between a device terminal and a gate signal. The net connection between a device terminal and a gate signal is topologically represented as a horizontal wire segment between a point and a vertical line. Let the coordinate of point A be (x_1, y_1) and the horizontal coordinate of line B be x_2. Then the distance between these two objects is

$$DST(A,B) = \left| x_1 - x_2 \right| \tag{6-3}$$

(3) Distance between two device terminals. Net connection between two device terminals A and B is topologically represented as a two-point wire segment W(A,B). If both of the two points locate at the same device row or at adjacent device rows, the distance functions are the same as equation 6-2 and 6-1. However, if these two points do not locate at adjacent rows, routing detour may occur if the insertion of extra vertical routing tracks is not allowed. Figure 6.8 shows two extreme cases of routing detours. Since only the unused terminal sites can be used as the through terminals, the net structure in Figure 6.8(a) requires a routing detour for a total length of 3L where L is the width of the device row. However, if through terminals are all aligned at the same vertical location as shown in Figure 6.8(b), no routing detour occurs. By weighting the factor of routing detour into the distance function, the distance function



(a)



(b)

Figure 6.8 The Routing Distance between Two Device Terminals. a) Worst Case; b) Best Case.

between two device terminals should be a weighted equation which gives the vertical part of the distance more weight. The weighted Manhattan distance between two device terminals in the Gate Matrix grid structure is given as

$$DST(A,B) = |x_1 - x_2| + |y_1 - y_2| + \frac{\alpha L}{2}|y_1 - y_2| \tag{6-4}$$

where L is the length of device row. The number of α is 1 if $|y_1 - y_2| \geq 2$ and is 0 otherwise.

6.4 Detailed Routing and Layout Improvement

Detailed routing is the task to find paths for the generated wire segments. Numerous routing techniques [65-79] have been proposed in the past three decades. In general, those routing algorithms fall into three categories, namely maze routers, line routers, and channel routers. In this dissertation, we use the channel routing technique [65-73] for several reasons:

(1) Channel routing is a simultaneous routing technique which usually results in shorter total wiring length.

(2) Gate Matrix structure allows the insertion of horizontal routing tracks which can be used as routing channels.

(3) Channel routing is a systematic routing method. Channel densities in both horizontal and vertical directions can be predicted before detailed routing process. Partial rerouting is also possible during the layout improvement process. Consequently, it is an ideal routing technique for

generating cells with different configurations.

6.4.1 Channel Assignment

The first step for a channel router is to assign each wire segment to the routing channels. If two ends of a wire segment are located at adjacent device rows, we assume that the wire segment will be realized in the routing channel which lies in between the two device rows. However, if the two ends are not located adjacently, the wire segment needs to be placed in several channels. The channel assignment process will be different in two different cases:

(1) The insertion of vertical routing tracks is allowed. When the insertion of vertical routing tracks is allowed, no routing detour is necessary. For the net structure in Figure 6.9, a vertical routing track can be inserted just beside the column of the starting net terminal. The connection between terminal A and B can always be realized by three zigzag line segments as shown in Figure 6.10.

(2) The insertion of vertical routing tracks is not allowed. In this case, we have to find a through terminal at each device row the wire segment passes through. The wire segment is then realized as a set of zigzag line segments as shown in Figure 6.11(a). Each line segment is assigned with the closest horizontal routing channel. In determining the through terminals on each device row, we use the shortest path algorithm [70] to find the set of line segments with the minimum total length. Figure 6.11(b) is an example of finding the through terminals for the non-adjacent wire segment.

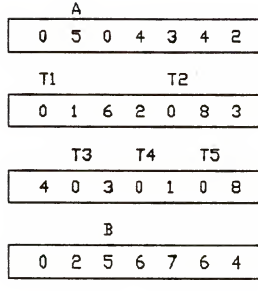


Figure 6.9 A Sample Net Structure.

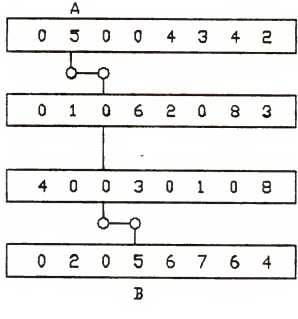


Figure 6.10 Channel Assignment by Inserting a Vertical Routing Track.

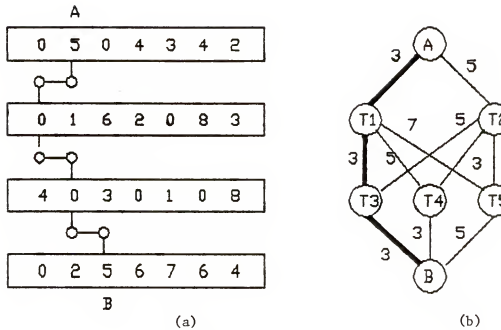


Figure 6.11 Channel Assignment by Shortest Path Algorithm. a) Decompose the Two-Point Wire into Several Short Wires; b) Select the Through Terminals Which Are Located on the Shortest Path.

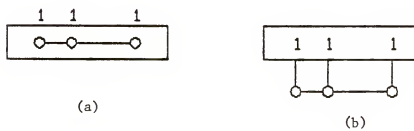


Figure 6.12 Single-Row Routing Patterns. a) Device Rows Are Used as Routing Channels; b) Use the Routing Channels between Two Device Rows.

6.4.2 Channel Routing

After all the wire segments are assigned to channels, the channel router executes routing on each channel. The channel routing technique implemented is similar to the one proposed in [66]. However, since the Gate Matrix layout structure has more than one kind of routing scheme, a combined technique is used to execute detailed routing.

Single-row routing. Gate Matrix layout has a large amount of internal nets of which the device terminals are located at the same device row. Those nets have the single-row routing pattern as shown in Figure 6.12. The algorithm used to solve the single-row routing problem is similar to the one proposed by Tsukiyama et al. [66]. All the wire segments with both ends located on the same device row are ordered by their left edges as shown in Figure 6.13. Those wires are then placed on the device row and the channel located below the device row. The routing result is shown in Figure 6.14.

Two-row routing. All the wire segments, whose both ends are located at adjacent device rows, have the two-row routing pattern as shown in Figure 6.15. The two-row routing problem is a classical channel routing problem and is solved by the constraint graph based [65] algorithm. As shown in Figure 6. 16, all the wire segments which have vertical routing constraints are solved by trunk partition (or called dogleging) [69] technique. The number of possible locations for trunk partition is restricted by the number of terminal sites on the device rows. However, insertion of vertical routing tracks can be used as the last resort to create more trunk partition sites.

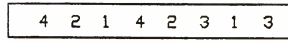


Figure 6.13 A Single-Row Net Structure.

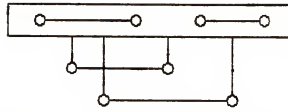


Figure 6.14 Routing by the Left-edge-first Algorithm.

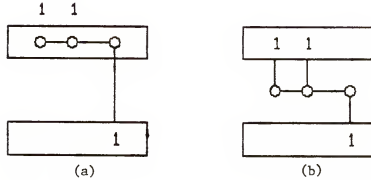


Figure 6.15 Two-Row Routing Patterns. a) Using Device Rows; b) Not Using Device Rows.

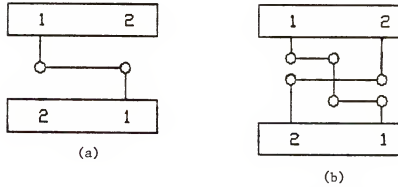


Figure 6.16 Solving Vertical Routing Conflicts by Trunk Partitions. a) Net 1 and Net 2 Cannot Be Realized at the Same Time; b) A Trunk Partition is Needed to Complete the Routing.

6.5 Layout Improvement

Several tactics are implemented to improve the cell layout after the routing process. Those improvement tactics emphasize either reducing the routing tracks or modifying the cell configuration but not both.

6.5.1 Reducing the Routing Tracks

There are two tactics which can be used in reducing the routing tracks.

(1) Fully utilize the area of device rows. In order to simplify the algorithm, device rows are not considered as part of the routing channel during the two-row routing process. However, available area on device rows can be fully utilized at the final layout improvement stage. The routing pattern shown as the left part in Figure 6.17 can be transformed into the corresponding patterns shown in the right side of the same figure. The transformation will reduce the number of horizontal routing tracks. Consequently, the electrical properties of the cell will be improved.

(2) Net transferring. As shown in Figure 6.18, the single row nets can be freely transferred from bottom channel to top channel or vice versa. This process will also help in reducing the number of horizontal routing tracks.

6.5.2 Modification of Cell Configuration.

As described in Chapter IV, the cell configuration is determined mainly by the number of device columns. However, for a specific device column. the cell configuration can also be slightly modified by trunk partition and routing detour.

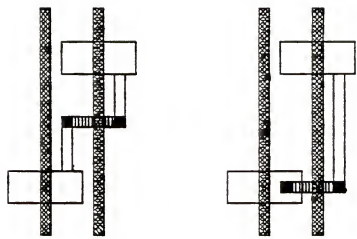


Figure 6.17 Layout Improvement by Fully Utilizing Device Areas.

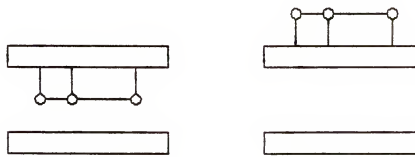


Figure 6.18 Net Transferring Technique for Layout Improvement.

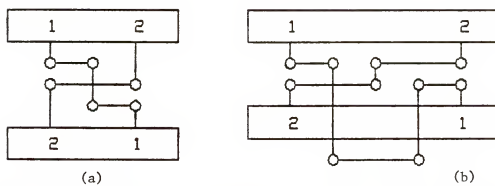


Figure 6.19 Modify Cell Configuration by Trunk Partition. a) One Trunk Partition; b) Three Trunk Partitions.

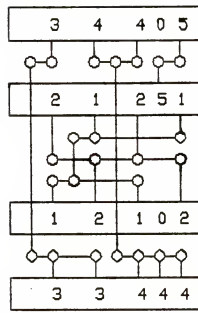
As shown in Figure 6.19(a), the dogleging routing schemes uses three horizontal routing tracks. However, as shown in Figure 6.19(b), the number of horizontal tracks can be reduced to two if extra trunk partitions are used. In general, in order to reduce one horizontal routing track, two more vertical routing tracks will have to be included. If the cell aspect ratio of a cell is originally

$$S = H/W = \frac{6(n_r + n_t)}{8n_c + 6n_v} \tag{6-5}$$

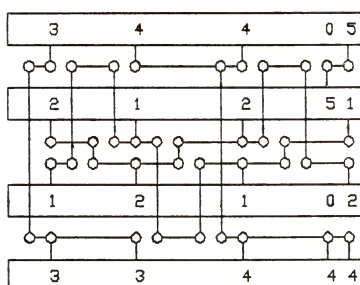
It will be

$$S' = H'/W' = \frac{6(n_r + n_t - 1)}{8n_c + 6(n_v + 2)} \tag{6-6}$$

after the cell modification process. Figure 6.20 shows a more sophisticated dogleging routing example of modifying the cell configuration.



(a)



(b)

Figure 6.20 Modification of Cell Configuration. a) Original Routing Scheme; b) Use a Lot of Trunk Partitions to Reduce the Number of Horizontal Routing Tracks.

CHAPTER VII
SUMMARY AND CONCLUSIONS

7.1 Experimental Results

The CMOS symbolic cell layout generation system has been implemented in C language on a Gould minicomputer under UNIX environment. The system, which still needs improvement, has shown two great advantages over the traditional manual design approach. The first one is that the design time for a typical 50 transistor cell has been reduced from several days to less than a minute. The short design turn-around-time has allowed the layout system to try a lot of cell configurations and select the one which has the best match with the chip floor plan. The second advantage for the system is that it emphasizes optimizing the total chip area instead of the individual cell area. Although the area for each generated cell may be larger than the corresponding manually designed cell, the total chip area is optimized through the reduction of the cell shape mismatch. Figure 7.1 and 7.2 demonstrates two computer generated cells based upon the cell boundary conditions shown in Figure 4.14(a) and Figure 4.14(c). As illustrated in Table 7.1, the area penalty for these two cells are 20% and 15% respectivey compared to the manually designed cells. However, the shape mismatches between the computer generated cells and the apportioned chip areas are generally lower than the manually designed cells. For the long-shape cell in Figure 7.1, the shape mismatch is 9% of the total cell area. For the wide-shape cell shown in Figure 7.2, the shape mismatch is 16% which is higher than the accepted level. The shape mismatch, however, can be reduced by the slight modification of the cell configuration. Figure 7.3 shows a different cell layout which is generated by modi-

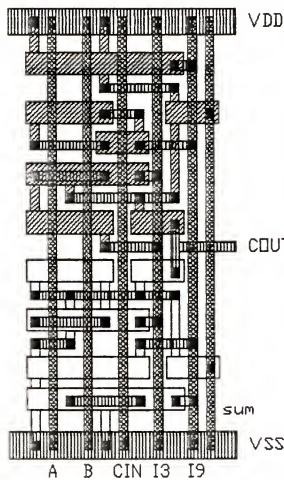


Figure 7.1 Computer Generated Fulladder Cell with Long Shape.

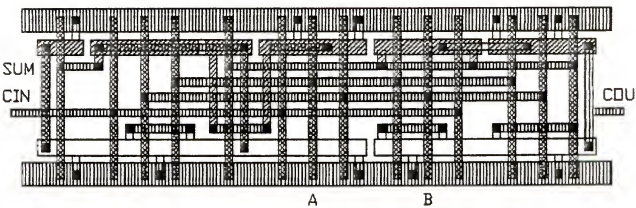


Figure 7.2 Computer Generated Fulladder Cell with Wide Shape.

Table 7.1 Geometric Information of the Computer Generated Cells

Layout	Width	Height	Cell Area	Area Penalty	Mismatch Area (%)
Figure 7.1	52	88	4576	20%	416 (9%)
Figure 7.2	154	46	7084	15%	1104 (16%)
Figure 7.3	124	54	6696	9%	496 (7%)

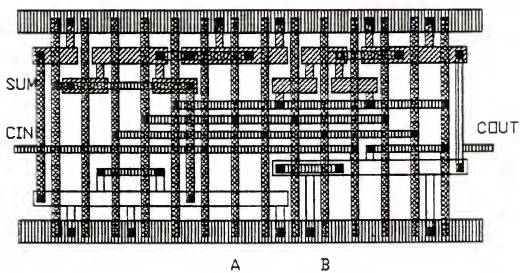


Figure 7.3 Computer Generated Cell Layout with a Configuration Slightly Different from the Cell in Figure 7.2.

fying the cell in Figure 7.2. After the layout improvement process, the shape mismatch is reduced from 16% to 7%. In general, the average area penalty for the computer generated cells is about 10% to 20% compared to manual design approach. Nevertheless, the area penalty for the total chip is below 10%. The reduction of total chip area is due to the flexibility of the automated cell layout generation system and the iterative improvements between the cell design and chip floor planning processes.

7.2 Summary

In this dissertation, we have presented a flexible CMOS cell design method to enhance the function of general cell layout system. The boundary requirements of the cell generation system were specified, and the necessary techniques for each task of the system were developed. In Chapter I, we addressed the need for a flexible cell generation system. With the growing use of general cell layout systems, we believe that research in this direction should be encouraged.

In Chapter II, we surveyed the research progress in the past ten years and found that flexible cell configuration is still a problem yet to be resolved.

In Chapter III, we investigated the complete automatic layout system for IC design and characterized the functional role of cell generation in the whole system. An integrated flexible cell generation system was then introduced which intend to perceive, characterize, and interpret the various physical characteristics of the practical cell layout problem.

In Chapter IV, we analyzed the boundary constraints for cell generation process. A grid structure, with a desired aspect ratio, was then constructed based upon the analysis.

In Chapter V, the focus was on device placement techniques which determined optimal device terminal locations on grid structure. A clustering technique was introduced for the purpose of an optimized ordering among the device columns. This technique determined the horizontal grid locations of the devices. Determination of the device orientations was based on the circuit heuristics and practical wiring considerations. A vertical constraint graph based technique was used to determine the vertical grid locations of the devices with an emphasis to reduce wiring detours.

In Chapter VI, the focus was on the development of routing techniques to realize the circuit connectivity. A terminal labeling technique was used to transform the circuit connectivity information from the circuit domain to the topological domain. Each multi-terminal net was then represented as a multi-point net on the grid structure. A revised minimum spanning tree algorithm was then implemented to decompose the multi-point net into a set of two-point wire segments. A combined techniques of channel routing and maze routing realized each wire segment with physical routing path.

7.3 Significance of This Research

The first accomplishment of this dissertation has been the extensive study of the real-time interaction between the chip floor planner and the cell generator. Although the study still needs further work, the experimental results have shown that the integration of the chip floor planner and the cell generator is a key factor in the success of an automatic IC layout system.

Our second accomplishment is to develop a cell structure model which links the symbolic layout domain and physical layout domain. Symbolic layout approach has a lot of advantages which include fast computation and tech-

nology independence. However, as most of the boundary conditions and cell specifications are given in terms of physical parameters, the cell structure model will help to integrate practical considerations into symbolic layout algorithms.

A third accomplishment is that the complex cell layout problem has been effectively divided into simpler subproblems. The simplicity of each subproblem results in the ability to characterize the various features of each individual subproblem. The "divide and conquer" method shows a practical way to solve the real world problem. Some of the innovative techniques developed for solving the subproblems should also be a useful reference for future research in this direction.

7.4 Recommendations for Future Research

In this dissertation, we have presented some results in flexible cell generation systems. Although there have been some encouraging results, the system is by no means complete. More research work is needed to enhance or improve the functions of the system. In the specific problem areas that we have addressed here, we recommend that further research be carried on the following areas:

1. Although the Gate Matrix Layout structure has proved to be a compact structure for cell layout, it still suffers some limitation in cell configuration. The minimum number of device columns is equal to the number of gate signals. It inhibits the generation of thin-shape cells. A new layout structure with more flexibility in cell configurations is needed to resolve the problem.
2. All the current chip floor planning algorithms assume that the cell area remains the same no matter how its configuration changes. How-

ever, the assumption has been proved incorrect based upon the experimental results of the cell generation system. Cell areas might differ greatly with different cell configurations. In searching for an optimized chip area, the chip floor planner need to consider the size variation problem together with the configuration problem. Research in this direction might be an important area in the future.

3. More electrical properties should be considered in cell layout besides the wiring length, the transistor abutment, and the cell configuration. It may, however, be impossible to optimize all the layout parameters at the same time. Further research in layout performance evaluation should be valuable.

In retrospect, it is clear that the design of silicon compiler to replace designers' work has faced many complex issues that are yet to be resolved. However, the accumulation of all the past accomplishments has put the "real" silicon compiler within reach.

After several years of research in this field, we came to believe that a solution to this problem exists. We hope that more researchers will take up this challenging problem in an effort to design a better chip in a much shorter turnaround time.

REFERENCES

[1] A. Feller, "Automatic Layout of Low-Cost Quick- Turnaround Random-Logic Custom LSI Devices," Proceedings of the 13th Design Automation Conference, pp. 79-85, 1976.

[2] G.P. Persky, D.N. Deutsch, and D.G. Schweikert, "LTX - A System for the Directed Automatic Design of LSI Circuits," Proceedings of the 13th Design Automation Conference, pp. 399-407, 1976.

[3] H. Beke, W. Sansen, "CALMOS: A Portable Software System for the Automatic and Interactive Layout of MOS/LSI," Proceedings of the 16th Design Automation Conference, pp. 102-108, 1979.

[4] J.E. Hassett, "Automatic Layout in ASHLAR: An Approach to the Problems of General Cell Layout for VLSI," Proceedings of the 19th Design Automation Conference, pp. 777-784, 1982.

[5] B.T. Preas and C.W. Gwyn, "Method of Hierarchical Automatic Layout of Custom LSI Circuit Masks," Proceedings of the 15th Design Automation Conference, pp. 206-211, 1978

[6] S. Muroga, *VLSI System Design -- When and How to Design Very-Large-Scale Integrated Circuits,* John Wiley and Sons, New York, 1982.

[7] K. Ueda, H. Kitazawa, and I. Harada, "CHAMP: Chip Floor Plan for Hierarchical VLSI Layout Design," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. CAD-4, No. 1, pp. 12-22, January 1985.

[8] U. Lauther, "A Min-Cut Placement Algorithm for General Cell Assemblies Based on a Graph Representation," Proceedings of the 16th Design Automation Conference, pp. 1-10, 1979.

[9] A.D. Lopez and H.F. Law, "A Dense Gate Matrix Layout Method for MOS VLSI," IEEE Journal of Solid-State Circuits, Vol. SC-15, No. 4, pp. 736-740, August 1980.

[10] A. Dunlop, "SLIP: Symbolic Layout of Integrated Circuits with Compaction," Computer Aided Design, Vol. 10, No. 6, pp. 387-391, November 1978.

[11] N. Weste, "Virtual Grid Symbolic Layout," Proceedings of the 18th Design Automation Conference, pp. 225-233, 1981.

[12] N. Weste, "MULGA -- An Interactive Symbolic Layout System for the Design of Integrated Circuits," Bell System Technical Journal, Vol. 60.

No. 6, pp. 823-858, July-August 1981.

[13] N. Weste and B. Ackland, "A Pragmatic Approach to Topological Symbolic IC Design," Proceedings of VLSI 81, pp. 117-129, 1981.

[14] B. Ackland and N. Weste, "An Automatic Assembly Tool for Virtual Grid Symbolic Layout," in *VLSI 83* , Elsevier Science Publishers, North Holland, pp. 457-468.

[15] C. Lursinsap and D. Gajski, "Cell Compilation with Constraints," Proceedings of the 21th Design Automation Conference, pp. 103-108, 1984.

[16] C.D. Rogers, J.B. Rosenberg, and S.W. Daniel, "MCNC's Vertically Integrated Symbolic Design System," Proceedings of the 22nd Design Automation Conference, pp. 62-68, 1985.

[17] J.B. Rosenberg, "Auto-Interactive Schematics to Layout Translation," Proceedings of the 22nd Design Automation Conference, pp. 82-87, 1985.

[18] Y.Z. Liao and C.K. Wong, "An algorithm to Compact a VLSI Symbolic Layout with Mixed Constraints," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. CAD-2, No. 2, pp. 62-69, April 1983.

[19] "ABCD: A Better Circuit Description Designer Reference," in *VIVID Systen User's Manual,* Section 19, Microelctronics Center of North Carolina, RTP, NC 27709, 1983.

[20] A. Weinberger, "Large Scale Integration of MOS Complex Logic: A Layout Method," IEEE Journal of Solid-State Circuits, Vol. SC-2, No. 4, pp. 182-190, December 1967.

[21] T. Ohtsuki, H. Mori, E.S. Kuh, T. Kashiwabara, and T. Fujisawa, "One-Dimensional Logic Gate Assignment and Interval Graphs," IEEE Transactions on Circuits and System, Vol. CAS-26, No. 9, pp. 675-684, September 1979.

[22] T. Uehara and W.M. vanCleemput, "Optimal Layout of CMOS Functional Arrays," IEEE Transactions on Computers, Vol. C-30, No. 5, pp. 305-312, May 1981.

[23] O. Wing, "Automated Gate Matrix Layout," Proceedings of the 1982 IEEE International Symposium on Circuits and Systems, pp. 681-685, 1982.

[24] J.T. Li, "Algorithms for Gate Matrix Layout," Proceedings of the 1983 IEEE International Symposium on Circuits and Systems, pp. 1013-1016, 1983.

[25] O. Wing, S. Hwang, and R. Wang, "Gate Matrix Layout," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,

Vol. CAD-4, No. 3, pp. 220-231, July 1985.

[26] N. Deo, M.S. Krishnamoorthy, and M.A. Langston, "Exact and Approximate Solutions for the Gate Matrix Layout Problem," IEEE Transactions on Computer-Aided Design of Integarted Circuits and Systems, Vol. CAD-6, No. 1, pp. 79-84, January 1987.

[27] N. Rose and J. Oldfield, "Printed-Wiring-Board Layout by Computers," Electronics and Power, pp. 373-380, October 1971.

[28] W.L. Engl, D.A. Mlynski, and Pernards, "Computer-Aided Topological Design for Integrated Circuits," IEEE Transactions on Circuit Theory, Vol. CT-20, No. 6, pp. 717-725, November 1973.

[29] M.C. Van Lier and R.H.J.M. Otten, "Automatic IC Layout: The Model and Technology," IEEE Transactions on Circuits and Systems, Vol. 11, pp. 845-854, November 1975.

[30] T.K. Ng and S.L. Johnsson, "Generation of Layouts from MOS Circuit Schematics A Graphic Theoretic Approach," Proceedings of the 22nd Design Automation Conference, pp. 39-45, 1985.

[31] W.M. vanCleemput, "Mathematical Models for the Circuit Layout Problem," IEEE Transactionas on Circuits and Systems, Vol. CAS-23, No. 12, pp. 759-767, December 1976.

[32] W. Wolf, J. Newkirk, R. Mathews, and R. Dutton, "DUMBO, A Schematic-To-Layout Compiler," in *Proceedings of the Third Caltech Conference on VLSI,* R. Bryant ed., Computer Science Press, Rockville, Md., pp. 379-394, 1983.

[33] R.P. Larsen, "Computer-Aided Preliminary Layout Design of Customized MOS Arrays," IEEE Transactionas on Computers, Vol. C-20, No. 5, pp. 512-523, 1971.

[34] P.W. Kollaritsch, and N.H.E. Weste, "Topologizer: An Expert System Translator of Transistor Connectivity to Symbolic Cell Layout," IEEE Journal of Solid-State Circuits, Vol. SC-20, No. 3, pp. 799-804, June 1985.

[35] M. Deo, *Graph Theory with Applications to Engineering and Computer Sciences,* Prentice-Hall, Englewood Cliffs, New Jersey, 1974.

[36] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design -- A System Perspective,* Addison-Wesley, Reading, Massachusetts, 1985.

[37] J. Hopcroft and R. Tarjan, "Efficient Planarity Testing," Journal of the Association for Computing Machinary, Vol. 21, No. 4, pp. 549-568, October 1974.

[38] M.A. Breuer, "Min-Cut Placement," Design and Fault- Tolerant Computing, Vol. 1, No. 4, pp. 343-362, October 1977.

[39] B.T. Preas and W.M. vanCleemput, "Placement Algorithms for Arbitrarily Shaped Blocks," Proceedings of the 16th Design Automation Conference, pp. 474-480, 1979.

[40] J.P. Blanks, "Near-Optimal Placement Using a Quadratic Objective Function," Proceedings of the 22nd Design Automation Conference, pp. 609-615, 1985.

[41] R. Otten, "Automatic Floorplan Design," Proceedings of the 19th Design Automation Conference, pp. 261-267, 1982.

[42] L. Shua and R.W. Dutton, "An Analytic Algorithm for Placement of Arbitrarily Sized Rectangular Blocks," Proceedings of the 22nd Design Automation Conference, pp. 602-608, 1985.

[43] S. Kirkpatrick, C.D. Gelatt, Jr., and M.P. Vecchi, "Optimization by Simulated Annealing," Science, Vol. 220, pp. 671-680, May 1983.

[44] R.H.J.M. Otten and L.P.P.P. van Ginneken, "Floorplan Design Using Simulated Annealing," Proceedings of International Conference on Computers and Design, pp. 96-98, November 1984.

[45] "A Unified Automated VLSI Design Environment," Technical Report, General Electric at Daytona Beach, Daytona Beach, Florida, April 1985.

[46] The EDIF Committee, *EDIF Specification, EDIF Electronic Design Interchange Format Version 110,* EDIF Administration Office, Mesa, Arizona, 1985.

[47] P.C. Gilmore, "Optimal and Suboptimal Algorithms for Quadratic Assignment problem," Journal of SIAM, Vol. 10, No. 2, pp. 305-313.

[48] E.L. Lawler, "The Quadratic Assignment problem," Management Science, Vol. 9, pp. 586-599, 1963.

[49] J.M. Kurtzberg, "On Approximation Methods for the Assignment Problem," Journal of ACM, Vol. 9, No. 4, pp. 419-439, October 1962.

[50] D. Shuler and E. Ulrich, "Clustering and Linear Placement," Proceedings of the 9th Design Automation Workshop, pp. 50-62, 1972.

[51] A. Hashimoto and J. Stevens, "Wire Routing by Optimizing Channel Assignment Within Large Apertures," Proceedings of the 8th Design Automation Workshop, pp. 155-168, 1971.

[52] E.S. Kuh, T. Kashiwahara, and T. Fujisawa, "On Optimum Singal-Row Routing," IEEE Transactions on Circuits and System, Vol. CAS-26, pp. 361-368, 1979.

[53] M. Bellmore and G.L. Nemhauser, "The Traveling Salesman Problem: A Survey," Journal of Operations Research, Vol. 16, pp. 538-558, 1968.

[54] M. Held and R.M. Karp, "A Dynamic Programming Approach Sequencing

Problems," SIAM Journal of Applied Mathematics, Vol. 10, pp. 196-210, 1962.

[55] J.D.C. Little, K.G. Murty, D.W. Sweeney, and C. Karel, "An Algorithm for the Traveling Salesman Problem," Journal of Operation Research, Vol. 11, pp. 979-989, 1963.

[56] D. Shapiro, "Algorithms for the Solution of the Optimal Cost Traveling Salesman Problem," Sc.D. Thesis, Washington University, St. Louis, 1966.

[57] G.T. Martin, "An Accelerated Euclidean Algorithm for Integer Linear Programming," in *Recent Advances in Mathematical Programming* , R.L. Graves and P. Wolfe Ed., McGraw-Hill, New York, pp. 311-318, 1963.

[58] S. Lin, "Computer Solution of the Traveling Salesman Problem," Journal of Operation, Bell System Technical Journal, Vol. 44, pp. 2245-2269, 1965.

[59] J.B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," Proceedings of the Ameican Mathematics Society, Vol. 7, pp. 48-50, 1956.

[60] H. Loberman and A. Weinberger, "Formal Procedures for Connecting Terminals with a Minimal Total Wire Length," Journal of ACM, Vol. 4, pp. 428-437, 1957.

[61] R.C. Prim, "Shortest Connection Networks and Some Generations," Bell System Technical Journal, Vol. 36, pp. 1389-1401, Nov. 1957.

[62] V.K.M. Whitney, "Algorithm 422: Minimal Spanning Tree," Communications ACM, Vol. 15, No. 4, pp. 273, April 1972.

[63] M. Hannan, "On Steiner's Problem with Rectinear Distance," SIAM Journal of Applied Mathematics, Vol. 14, pp. 255-265, 1966.

[64] E.N. Gilbert and H.O. Pollak, "Steiner Minimal Trees," SIAM Journal of Applied Mathematics, Vol. 16, pp. 1- 29, 1968.

[65] T. Yoshimura and E.S. Kuh, "Efficient lgorithms for Channel Routing," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. CAD-1, No. 1, pp. 25-35, January 1982.

[66] S. Tsukiyama, E.S. Kuh, and I. Shirakawa, "An Algorithm for Single-Row Routing with Prescribed Street Congestions," IEEE Transactions on Circuits and Systems, Vol. CAS-27, No. 9, pp. 765-771, September 1980.

[67] R. Raghavan and S.K. Sahni, "The Complexity of Single Row Routing," IEEE Transactions on Circuits and Systems, Vol. CAS-31, No. 5. pp. 462-472, May 1984.

[68] B. Ting, E.S. Kuh, and I. Shirakawa, "The Multilayer Routing Problem: Algorithms and Necessary and Sufficient Conditions for the Single-Row

Single-Layer Case," IEEE Transactions on Circuits and Systems, Vol. CAS-23, pp. 768-778, 1976.

[69] D.N. Deutch, "A Dogleg Channel Router," Proceedings of the 13th Design Automation Conference, pp. 425-433, 1976.

[70] K. Sato, H. Shimoyama, T. Naya, M. Ozaki, and T. Yahara, "A Grid-Free Channel Router," Proceedings of the 17th Design Automation Conference, pp. 22-31, 1980.

[71] I. Skirakawa, "Routing for High Density Printed Wiring Boards," in *Hardware and Softawre Concepts in VLSI,* G. Rabbat Ed., Van Nostrand Reinhold Company Inc., New York, pp. 452-479, 1983.

[72] S. Tsukiyama, S. Shirakawa, and S. Asahara, "An Algorithm for the Via Assignment Problem in Multilayer Backboard Wiring," IEEE Transactions on Circuits and Systems, Vol. CAS-26, No. 6, pp. 369-377, 1979.

[73] A Consideration of the Number of Horizontal Grids Used in the Routing of a Masterslice Layout," Proceedings of the 19th Design Automation Conference, pp. 121-128, 1982.

[74] S. Akers, "Routing Techniques," in *Design Automation of Digital Systems: theory and Techniques,* M.A. Breuer Ed., Prentice-Hall, Englewood Cliffs, New Jersey, pp. 282-333, 1972.

[75] D.W. Hightower, "The Interconnection Problem: A Tutorial," Computer, No. 7, pp. 18-32, 1974.

[76] C.Y. Lee, "An Algorithm for Path Connections and Its Applications," IRE Transactions on Electronic Computers, Vol. EC-10, pp. 346-365, 1961.

[77] S.B. Akers, "A Modification of Lee's Path Connection Algorithm," IEEE Transactions on Electronic Computers, Vol. EC-16, pp. 97-98, 1967.

[78] D.W. Hightower, "A Solution to Line Routing Problems on the Continuous PLane," Proceedings of the 6th Design Automation Workshop, pp. 1-24, 1969.

[79] R. Mattison, "A High Quality Low Cost Router for LSI/VLSI," Proceedings of 9th Design Automation Workshop, pp. 94-103, 1972.

BIOGRAPHICAL SKETCH

Wanhao Li was born in Taichung, Taiwan, Republic of China, on June 4, 1955. In September, 1972, he was admitted to National Taiwan University, where in June, 1976, he received a bachelor's degree in electrical engineering. In August 1981, he was awarded a one-year fellowship from Lamar University, and he received the Master of Engineering in August, 1982. He started his Ph.D. studies in electrical engineering at the University of Florida in August 1983. He has been serving as a research assistant at the Center for Information Research since September, 1983.

He plans to join the Motorola Inc. in July, 1987, where he will continue research on CAD for VLSI.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Julius T. Tou, Chairman
Graduate Research Professor of
Electrical Engineering and
Computer and Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

John Staudhammer
Professor of Electrical Engineering and
Computer and Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Fred J. Taylor
Professor of Electrical Engineering and
Computer and Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Yuan-Chieh Chow
Associate Professor of
Computer and Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Mark C. Yang
Professor of Statistics

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

August 1987

Dean, College of Engineering

Dean, Graduate School